DOI: 10.14750/ME.2026.038

University of Miskolc

Faculty of Mechanical Engineering and Informatics

Efficiency Analysis and Optimization of Concept Lattice Reduction Methods

Ph.D. Dissertation

József Hatvany Doctoral School of Information Science, Engineering and Technology

Research Area
Applied Computer Science

Research Group
Data and Knowledge Bases

Author:
Mohammed Ali Daash Alwersh
M.Sc. in Computer Science

Head Of Doctoral School:
Prof. Dr. Jenő Szigeti

Academic Supervisor:
Prof. Dr. László Kovács

Miskolc, Hungary 2025

DOI: 10.14750/ME.2026.038

Declaration

The author hereby declares that this thesis has not been submitted, either in the same or in a different form, to this or to any other university for obtaining a PhD degree. The author affirm that the submitted work is his own and the appropriate credit has been given where reference has been addressed to the work of others.

Miskolc, 2025.

Mohammed Ali Daash Alwersh

DOI: 10.14750/ME.2026.038

Acknowledgments

I am deeply grateful to my supervisor, Prof. Dr. László Kovács, for his ongoing support, guidance, and encouragement throughout the past years. His invaluable insights, patience, and expertise have significantly contributed to the development of this dissertation. This work is a reflection of his mentorship as well as the sustained effort and dedication invested over the past four years.

I am also profoundly grateful to the faculty and staff of the Faculty of Mechanical Engineering and Informatics at the University of Miskolc, within the Research Field of Applied Computer Science, for providing the necessary resources and fostering an environment conducive to academic excellence. Their support and expertise have greatly enriched my research experience. I also acknowledge the financial support provided by Stipendium Hungaricum (SH), which helped me successfully complete my research and focus on the academic pursuits essential to this dissertation.

I extend my heartfelt appreciation to my family, my sons, Mustafa Alwersh and Kadhim Alwersh, and special thanks to my wife, Maryam Alwersh, for her unwavering support and for standing by me every step of the way to achieve my goal. Despite the challenges and difficulties I encountered, her constant support and encouragement have been a steadfast source of motivation, helping me navigate the inevitable ups and downs of the research process.

Thank you all for your contributions and for being an integral part of this academic endeavor.

Mohammed Ali Daash Alwersh

DOI: 10.14750/ME.2026.038

Table of Contents

DOI: 10.14750/ME.2026.038

DOI: 10.14750/ME.2026.038

List of Figures

DOI: 10.14750/ME.2026.038

List of Tables

DOI: 10.14750/ME.2026.038

List of Abbreviations

FCA	Formal Concept Analysis
CL	Concept Lattice
KDL	K-Means Dijkstra on Lattice
KVL	K-Means Vector on Lattice
KCS	Kernel Concept Set
ICFCA	International Conference on Formal Concept Analysis
ICCS	International Conference on Conceptual Structures
CLA	Concept Lattices and their Applications
SVD	Singular Value Decomposition
CbO	Close-by-One
FcbO	Fast Close-by-One
RDF	Resource Description Framework
DM	Data Mining
OWL	Web Ontology Language
SparQL	SPARQL Protocol and RDF Query Language
IR	Information Retrieval
CC	Conceptual Clustering
DBI	Davies-Bouldin Index
CS	Concept Similarity
EMO-CC	Multiobjective Evolutionary Conceptual Clustering Methodology
KDD	Knowledge Discovery in Databases
GA	Genetic Algorithm
SA	Simulated Annealing
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing

DOI: 10.14750/ME.2026.038

Preface

This dissertation embodies a focused endeavor in the realm of knowledge engineering, particularly at the intersection of data mining and Formal Concept Analysis (FCA). Since its inception in the early 1980s by Rudolf Wille, FCA has been recognized as a powerful mathematical tool for representing and analyzing the relationships between objects and attributes within formal contexts. By structuring information into concept lattices, hierarchical diagrams that capture the relationships between objects and attributes, FCA facilitates the discovery of meaningful patterns in diverse fields, including software engineering, information retrieval, e-learning systems, bioinformatics, and beyond.

Yet, as datasets expand in size and complexity, the concept lattices derived from them can grow exponentially, posing formidable computational and interpretive challenges. Traditional FCA methods, while theoretically elegant, often become computationally intensive and cognitively overwhelming, hindering the effective utilization of these structures in large-scale data analytics. This dissertation addresses these challenges head-on through a series of three integrated contributions, each representing a strategic step toward more scalable, efficient, and human-centered FCA methodologies.

A key groundwork is first laid out, establishing several foundational pillars that guide the methods proposed here. These include the notion of kernel concepts, specially chosen concepts that serve as anchors for understanding and reducing a concept lattice, alongside an asymmetrical distance metric that adapts Dijkstra's algorithm for cost-aware navigation. A baseline greedy framework for concept selection further sets the stage for the more specialized methods and cognitively aligned reduction strategies that follow.

The first contribution introduces two novel extensions of the k-means algorithm, K-Means Dijkstra on Lattice (KDL) and K-Means Vector on Lattice (KVL), to adapt clustering-based reduction strategies for FCA. KDL leverages the inherent hierarchical structure of categorical data by incorporating a graph-based distance measure derived from FCA. This ensures that reductions remain faithful to the underlying conceptual relationships, yielding more interpretable and structurally consistent lattices. In contrast, KVL transforms formal concepts into numerical vectors, allowing the application of conventional k-means clustering at scale. While this vectorization simplifies complexity and improves computational efficiency, careful consideration is given to preserving lattice quality. Together, KDL and KVL mark an initial leap toward practical, data-driven lattice reduction that balances complexity management with interpretability.

Building on these foundations, the second contribution, the Kernel Concept Set (KCS) approach proposes a frequency- and cost-based strategy for selecting a core subset of concepts. By determining a kernel that covers the most critical and frequently occurring attributes, KCS optimizes reduction while maintaining essential structure. This approach goes beyond the first step's clustering-centric views, providing a more refined selection mechanism that directly addresses the trade-off between completeness and efficiency.

DOI: 10.14750/ME.2026.038

Finally, the third contribution introduces cognitive and linguistic strategies for scalable concept lattice reduction. Inspired by human language optimization principles, this model employs a finite "vocabulary" of high-frequency conceptual units (kernel concepts) and an injective mapping function to ensure each concept is represented uniquely and meaningfully. By integrating Genetic Algorithms and Simulated Annealing alongside a learning-based module, the model identifies an optimal kernel subset that minimizes total generation cost, a measure reflecting both computational and cognitive resources. This interdisciplinary approach not only reduces lattice size but also aligns the resulting structures with human cognitive processes, making the reduced lattices both computationally feasible and intuitively comprehensible.

Collectively, these three contributions form a coherent research trajectory. Starting from harnessing clustering methods for initial complexity control (KDL and KVL), moving through a frequency- and cost-informed selection of pivotal concepts (KCS), and culminating in a linguistically and cognitively oriented optimization framework, this dissertation offers a comprehensive toolkit for addressing the scalability, efficiency, and interpretability challenges inherent in FCA.

By fusing computational heuristics, cognitive insights, and linguistic principles into the FCA reduction process, this work advances FCA from a theoretically compelling method to a practical, user-aligned analytical framework. It lays the groundwork for broader adoption of FCA in large-scale data analysis, equipping researchers and practitioners with strategies to navigate, understand, and ultimately derive more meaningful insights from complex and voluminous data

DOI: 10.14750/ME.2026.038

Chapter 1: Introduction

1.1. Research Context

In recent decades, the exponential growth of data across diverse domains, from healthcare and finance to e-learning and social media, has necessitated increasingly sophisticated methods to extract, represent, and interpret meaningful patterns. The convergence of data mining, machine learning, and knowledge engineering has driven researchers and practitioners to seek frameworks that not only handle vast amounts of information efficiently but also facilitate human understanding of underlying structures and relationships. Among these frameworks, Formal Concept Analysis (FCA) [1] has emerged as a mathematically rigorous and conceptually rich approach to organizing and interpreting complex datasets.

FCA operates by mapping data described as objects and attributes into a conceptual hierarchy known as a concept lattice. This lattice encodes the inherent relationships within the data, revealing clusters of attributes that co-occur among sets of objects. The resulting structure is more than just a visualization tool; it serves as a knowledge representation mechanism that can inform decision-making, discovery of patterns, and the identification of subtle dependencies. However, as data complexity intensifies, due to high dimensionality, large numbers of objects, or the intricate interplay of attributes, the concept lattices constructed from such datasets often become prohibitively large and complex. Despite their theoretical elegance, these massive lattices pose practical challenges: they demand substantial computational resources to construct and manipulate, and they may overwhelm human analysts attempting to derive insights. The problem is compounded by the fact that many fields relying on FCA, such as bioinformatics or software engineering, frequently involve large-scale and evolving datasets [2].

In light of these considerations, the need for concept lattice reduction methods becomes evident. By judiciously streamlining the concept lattice to retain essential structural and informational properties while discarding redundancies and less critical elements, these reduction methods pave the way for more efficient analysis and clearer interpretability. Achieving a balance between lattice complexity and informational fidelity is a non-trivial challenge, especially as reduction techniques must ensure that critical patterns and relationships remain intact for meaningful analysis.

The quest for more efficient and interpretable FCA-based frameworks does not exist in isolation. The broader landscape of knowledge engineering and data analytics is also grappling with scalability and accessibility issues. Just as natural language processing research has evolved to manage complexities of informal language on social media [3], and intelligent tutoring systems have integrated sophisticated knowledge models to adapt learning materials [4], [5], so too must FCA methodologies evolve. These parallel efforts underscore a universal trend: as data grows in volume and complexity, analytical approaches must become more adaptive, intelligent, and scalable.

This dissertation seeks to tackle these intertwined issues of scalability and interpretability in FCA. It does so by developing and refining foundational pillars for lattice reduction, chief among them kernel concepts (centrally important nodes in a lattice) and cost-aware distance metrics and building upon these to propose multiple specialized reduction strategies. The overarching ambition is to fortify FCA's practical utility in handling large-scale data,

DOI: 10.14750/ME.2026.038

transforming an elegant theoretical framework into a truly accessible tool for knowledge discovery in complex domains.

Building upon this motivation, the dissertation advances three central contributions. First, it introduces clustering-based reduction strategies that adapt the K-means paradigm to FCA, namely KDL and KVL, which capture structural and attribute-based proximities in the lattice. Second, it develops the kernel concept framework as an original idea for selecting a strategically small yet influential subset of concepts that minimize derivation cost while preserving interpretability. Third, it proposes an optimization approach based on genetic algorithms, enhanced with neural-network-assisted evaluation, to efficiently mine kernel concepts and extend their applicability to domains such as computational linguistics. Collectively, these contributions establish a coherent methodological foundation for scalable and interpretable FCA, while also opening avenues for broader applications in large and complex data environments.

1.2. Problem Statement

FCA provides a mathematically sound and conceptually intuitive framework for representing complex data through concept lattices hierarchical structures that reveal intricate relationships between objects and attributes. While FCA has demonstrated considerable value in various domains, from knowledge engineering and intelligent tutoring systems to social media analysis and sentiment mining, its practical application is often hampered by a critical and persistent issue: the exponential growth in the number of formal concepts and, consequently, the size and complexity of the resulting concept lattice.

As datasets become more extensive, heterogeneous, and dynamic, the concept lattices derived from them can become prohibitively large and unwieldy. This exponential complexity leads to significant computational overheads in lattice construction, maintenance, and navigation. It also creates formidable interpretability challenges. Analysts, domain experts, and automated reasoning tools struggle to extract meaningful insights from a lattice that is both visually and structurally dense, rife with redundancies, and difficult to navigate.

Although several reduction techniques have attempted to mitigate these challenges by pruning less relevant concepts, applying frequency-based filters, or introducing abstraction mechanisms to simplify the lattice structure, they often suffer from critical limitations. A major issue is computational inefficiency, since many reduction algorithms do not scale well and result in prohibitive runtime and memory consumption, particularly for large or evolving datasets. Another challenge is the inadequate balance between complexity and fidelity, as some approaches oversimplify the lattice and discard critical information, thereby undermining the reliability of subsequent analyses. Moreover, while certain strategies reduce lattice size, they do not sufficiently enhance interpretability or improve cognitive accessibility, leaving users grappling with opaque and dense structures. Finally, many of the proposed approaches remain fragmented, lacking a unifying framework capable of integrating computational optimization with cognitive and linguistic strategies as well as systematic selection criteria for core concepts. Consequently, practitioners are often forced to rely on ad hoc or domain-specific solutions that do not generalize well across different datasets or application domains.

This gap in the literature, where scalability, efficiency, interpretability, and adaptability to varying contexts remain only partially addressed, represents the crux of the problem. It underscores the urgent need for holistic, optimized reduction techniques that not only

DOI: 10.14750/ME.2026.038

streamline concept lattices but also retain their informational richness and align more closely with human cognitive processes.

In essence, the challenge is to develop robust, scalable, and cognitively aligned concept lattice reduction methodologies that fulfill multiple objectives simultaneously: to significantly improve computational performance, to maintain or enhance interpretability, to preserve essential relationships and data patterns, and to integrate seamlessly into diverse application scenarios. This dissertation aims to tackle this core problem head-on, proposing innovative solutions, ranging from specialized k-means-based lattice clustering algorithms and kernel concept set selection to cognitive and linguistic optimization frameworks, that collectively advance the state of the art in FCA-based data analysis.

1.3. Research Objectives

The overarching aim of this research is to advance concept lattice reduction in FCA by making it more computationally efficient, scalable, and aligned with human interpretive processes. In pursuit of this aim, the research is guided by the following objectives.

First, it seeks to assess the limitations of existing reduction methods by thoroughly examining current approaches to concept lattice reduction and identifying their shortcomings in terms of scalability, computational efficiency, and interpretability. This evaluation highlights gaps in the literature and informs the strategic direction for new methodologies. Building on this, the second objective is to enhance computational efficiency and scalability by developing and refining reduction techniques that significantly decrease the time and resource requirements for constructing and managing concept lattices, thereby enabling the application of FCA to large-scale, high-dimensional datasets.

A third objective is to preserve structural integrity and informational fidelity, ensuring that the proposed reduction methods maintain essential hierarchical relationships and key data patterns within the lattice so that reduced structures remain meaningful representations of the underlying dataset. Closely related to this, the fourth objective is to improve interpretability and cognitive alignment by applying principles inspired by human language optimization to identify a minimal yet expressive subset of core concepts. This streamlines the lattice, making it more accessible and understandable, ultimately enhancing usability for analysts.

The fifth objective is to establish a unified and adaptable framework for reduction techniques, integrating various strategies into a cohesive system that allows flexible adjustments based on domain-specific needs, data characteristics, or interpretive goals. Such a unified perspective supports systematic exploration and tuning of different approaches. Finally, the sixth objective is to empirically validate and benchmark the proposed methods through rigorous experiments, standardized evaluation metrics, and representative datasets, demonstrating their effectiveness, versatility, and relevance across multiple application scenarios.

By achieving these objectives, the research aims to transform FCA from a theoretically appealing but computationally intensive approach into a more agile, interpretable, and widely applicable framework for knowledge representation and data-driven decision-making.

DOI: 10.14750/ME.2026.038

1.4. Research Questions

Building upon the problem statement and research objectives, this study seeks to address key questions that probe into the theoretical and practical dimensions of concept lattice reduction in FCA. The research questions are framed to guide the investigation towards more efficient, interpretable, and integrative reduction methodologies:

1. What are the limitations of current concept lattice reduction methods in terms of computational efficiency, scalability, and interpretability, and how do these constraints hinder their widespread adoption in real-world scenarios?
2. How can reduction techniques be optimized or reimagined to handle increasingly large and complex datasets without imposing prohibitive computational costs, thereby making FCA a more viable option for big data contexts?
3. In the process of simplifying the lattice, how can essential hierarchical relationships and the core informational content be preserved, ensuring that reduced lattices remain faithful, reliable representations of the underlying data?
4. How can concepts inspired by human linguistic efficiency be employed to identify a minimal yet expressive set of core concepts, thereby enhancing the interpretability and cognitive accessibility of the resulting reduced lattice?
5. Can diverse reduction strategies, including kernel concepts identification, and cognitively informed models, be integrated into a unified framework that allows flexible adaptation across various data domains, analytical objectives, and resource constraints?
6. How can the performance and utility of the proposed reduction techniques be rigorously evaluated against standardized metrics and representative datasets, and to what extent can these methods be generalized across different application areas?

1.5. Significance of the Study

This research is significant as it enhances both the theoretical and practical dimensions of Formal Concept Analysis. Theoretically, it introduces refined reduction methodologies that address longstanding challenges of computational complexity and interpretability, thereby advancing the core understanding of FCA's scalability. Practically, by producing more manageable and cognitively accessible lattices, the work broadens FCA's usefulness across various domainsranging from knowledge management to data-driven decision-making, enabling clearer insights from large and complex datasets.

1.6. Scope and Limitations

The scope of this research encompasses the development, integration, and empirical evaluation of concept lattice reduction techniques within the framework of FCA. The focus lies on enhancing computational efficiency, ensuring interpretability, and retaining essential structural properties of the lattice. The study involves testing various datasets and employing standardized performance metrics to validate proposed methodologies. However, certain limitations apply. The research does not aim to cover all possible data types or application domains, and the selection of evaluation metrics may not capture every facet of reduction quality. Moreover, while the proposed methods strive for broad applicability, domain-

DOI: 10.14750/ME.2026.038

specific customization may still be required. These constraints acknowledge the complexity and evolving nature of data challenges, guiding realistic expectations for the results.

1.7. Thesis Structure

This dissertation is organized into eight chapters. Chapter 1 introduces the research context, outlining the problem, objectives, significance, scope, and key research questions. Chapter 2 lays the theoretical foundation of FCA, defining formal contexts and concepts, discussing various algorithmic strategies, and addressing challenges in managing large-scale data with lattice reduction methods. In Chapter 3, a literature review examines existing techniques for reducing concept lattices, such as redundancy removal and clustering-based strategies, highlighting the need for novel approaches that balance scalability with interpretability. Chapter 4 presents the core principles of our proposed methods, introducing kernel concepts, a cost-based distance metric adapted from Dijkstra's algorithm, and a baseline greedy selection process to underpin our advanced reduction strategies. Chapter 5 introduces two clustering algorithms, KDL and KVL adapted from K-Means for FCA, detailing their theoretical bases and experimental evaluations on synthetic and real-world datasets. Building on this, Chapter 6 proposes the Kernel Concept Set (KCS) method, which leverages frequency metrics and derivation costs to identify pivotal concepts, thereby reducing complexity while preserving structural relationships. Chapter 7 further refines lattice reduction by incorporating heuristic and machine-learning approaches, such as Genetic Algorithms and Simulated Annealing, to optimize kernel concept selection while ensuring human-aligned representations. Finally, Chapter 8 concludes the dissertation by summarizing the primary achievements and outlining future research directions.

DOI: 10.14750/ME.2026.038

Chapter 2: Foundations of Formal Concept Analysis

2.1. Overview

FCA was introduced by Rudolf Wille in 1982 as a specialized subfield emerging from applied mathematics, grounded in the notions of "concept" and "concept hierarchy." Over time, FCA has evolved into an unsupervised machine learning approach adept at uncovering and representing conceptual structures embedded within data. Its initial mathematical underpinnings have broadened FCA's appeal, making it well-known, particularly in computer science, though its influence also extends to fields such as data mining [6], [7], knowledge representation [8], information management [9], and beyond. Since its inception, FCA has inspired hundreds of scholarly publications and has been supported by foundational texts, most notably the work of Ganter and Wille [10] provided the mathematical foundation of FCA, as well as key volumes by Davey and Priestley [11]. Historically, the Darmstadt research group in Germany played a pivotal role in FCA's early development, and today, FCA research spans the globe, supported by annual international conferences including the International Conference on Formal Concept Analysis (ICFCA), the International Conference on Conceptual Structures (ICCS), and the Concept Lattices and their Applications (CLA) meeting series. FCA's diverse range of applications now includes not only computer science but also statistics, applied mathematics, medicine, psychology, social science [12], [13], [14], artificial intelligence, and information retrieval [9].

At its core, FCA provides a methodology for analyzing binary data, where data is represented by objects and attributes, and uncovering the fundamental patterns, dependencies, and implications present in this data. In practice, FCA takes a binary context (a set of objects and their associated attributes) as input and produces sets of "natural clusters" of objects and attributes as output. These conceptual clusters can then be visually represented as a Hasse diagram (or line diagram), also known as a concept lattice or Galois lattice. This lattice representation reveals the intrinsic structural relationships concealed within the binary data. By offering a graphical and conceptual viewpoint, FCA enables clearer and more meaningful interpretations of complex datasets. Essentially, from a given collection of objects and attributes, FCA facilitates the extraction of a relevant ontology a systematic, philosophically grounded representation of entities and their interrelations enhancing the transparency and informational value of the data under study. For an in-depth exploration of FCA's role in knowledge discovery and information science, readers are directed to a detailed survey available in [15].

2.2. Structure of Concept Lattices

In this section, we establish the fundamental notions underlying FCA, starting with the definition of a formal context and moving toward the concept lattice that emerges from it. We introduce key elements such as formal concepts, which pair sets of objects and attributes, and explain how these concepts form a concept lattice that reveals the inherent structure of the data. To ground these ideas, we explore the mathematical constructs that support FCA, including Galois connections and closure operators, along with their essential properties. This theoretical foundation paves the way for understanding how concepts relate to one

DOI: 10.14750/ME.2026.038

another and how the lattice embodies a complete and well-organized representation of the given dataset.

2.2.1. Data Representation as Input

At the heart of FCA lies a binary data representation often called a cross-table or incidence table. This table links a set of objects (usually placed along rows) to a set of attributes (usually placed along columns). Each cell in the table marks whether a given object possesses a particular attribute, commonly indicated by a symbol (e.g., "×"). If the cell is empty, the object does not have that attribute [10].

As a simple illustration, consider a collection of laptop models (objects) and a selection of their attributes (characteristics) as shown in Table 2.1. Suppose we have four laptops L_1, L_2, L_3, L_4, and four attributes: "Touchscreen" (T), "Backlit Keyboard" (B), "Solid-State Drive (SSD)" (S), and "Detachable Screen" (D). We might arrange them as follows:

Table 2.1. Cross Table

	T	**B**	**S**	**D**
L_1	×	×	×	
L_2		×	×	
L_3	×		×	×
L_4	×		×	

In this example, the symbol "×" in the cell for L_1 and T means L_1 (Laptop 1) has a touchscreen, whereas the blank cell for L_2 and T means L_2 does not have a touchscreen. By capturing objects and attributes in this manner, FCA can methodically derive formal concepts and the resulting lattice that reveals the underlying conceptual structure within the dataset.

Definition 2.1 (Formal Context):

A formal context is a triple (G, M, I) where G is a non-empty set of objects, M is a non-empty set of attributes, and $I \subseteq G \times M$ is a binary relation. If $(g, m) \in I$, it indicates that the object $g \in G$ has attribute $m \in M$. Each "×" in the cross-table corresponds to a pair $(g, m) \in I$ [10].

In essence, a cross-table provides a straightforward and intuitive representation of data suitable for applying FCA. From this foundation, one can extract the conceptual structures known as formal concepts and arrange them into a concept lattice, thereby uncovering and visualizing meaningful patterns and relationships in the data.

2.2.2. Operators for Concept Formation

From each formal context (G, M, I), where G is a non-empty set of objects, M is a non-empty set of attributes, and $I \subseteq G \times M$ is the incidence relation, we derive two fundamental operators that map subsets of objects to subsets of attributes and vice versa. These are known as the concept forming operators, and they are central to identifying the formal concepts that constitute a concept lattice.

Definition 2.2 (Concept Forming Operators):

DOI: 10.14750/ME.2026.038

Consider a formal context (G, M, I). For any subset of objects $X \subseteq G$ and any subset of attributes $Y \subseteq M$, define:

$$X^{\uparrow} = \{m \in M \mid \forall g \in X, (g, m) \in I\},$$

$$Y^{\downarrow} = \{g \in G \mid \forall m \in Y, (g, m) \in I\}.$$

In other words, $X^{\uparrow}$ is the set of attributes common to every object in X, and $Y^{\downarrow}$ is the set of all objects that have every attribute in Y [10].

Remarks:

- The operator $(\cdot)^{\uparrow}$maps subsets of objects in G to subsets of attributes in M. Intuitively, if $X \subseteq G$, then $X^{\uparrow}$ returns the intersection of attributes shared by all objects in X.
- The operator $(\cdot)^{\downarrow}$ is its dual: it takes subsets of attributes $Y \subseteq M$ and returns all objects in G that possess all attributes in Y.

Example 2.1:

Recall our earlier example with laptops as objects and their features as attributes. Let:

- $G = \{L_1, L_2, L_3, L_4\}$ represent four laptop models.
- $M = \{T\ (Touchscreen),\ B\ (Backlit\ Keyboard),\ S\ (\text{SSD}),\ D\ (Detachable\ Screen)\}$.
- The relation I (indicated by "×") is given as shown in the formal context in Table 2.1:

From this context:

1 Consider $X = \{L_1, L_3\}$. L_1 has $\{T, B, S\}$, and L_3 has $\{T, S, D\}$. The attributes common to both L_1 and L_3 are T and S, so $\{L_1, L_3\}^{\uparrow}$=$\{T, S\}$.

2 Consider a single object set $X = \{L_4\}$. L_4 has $\{T, S\}$. Thus, $\{L_4\}^{\uparrow}$= $\{T, S\}$.

For attributes:

- Let $Y = \{B, S\}$. We want all objects that have both a Backlit Keyboard and a Solid-State Drive. L_1 and L_2 fit this description, hence $\{B, S\}^{\downarrow} = \{L_1, L_2\}$.
- Let $Y = \{S\}$. All laptops with a Solid-State Drive are L_1, L_2, L_3, and L_4, so $\{S\}^{\downarrow} = \{L_1, L_2, L_3, L_4\}$.

These concept-forming operators are crucial building blocks. By capturing which attributes characterize a set of objects, and which objects exhibit a particular combination of attributes, they enable us to define and understand formal concepts. Ultimately, these concepts can be combined to form a concept lattice, a structured representation that reveals intricate relationships within the data.

DOI: 10.14750/ME.2026.038

2.2.3. From Formal Concepts to the Concept Lattice

A central pillar of Formal Concept Analysis is the notion of formal concepts. These represent well-defined clusters of objects and attributes derived from the given formal context, embodying the intuitive idea of concepts, each concept corresponds to a group of objects sharing a precisely matching set of attributes [10].

Definition 2.3 (Formal Concept):

Given a formal context (G, M, I), a formal concept is a pair (X, Y) with $X \subseteq G$ and $Y \subseteq M$ such that:

$$X^{\uparrow} = Y, Y^{\downarrow} = X.$$

In other words, (X, Y) forms a formal concept if and only if X contains just objects sharing all attributes from Y and Y contains just attributes shared by all objects from X, with no extraneous elements. Here, X is called the extent and Y is called the intent of the concept. Extents are precisely the objects that share all attributes in Y, and the intent Y represents all attributes that these objects X have in common.

Example 2.2:

Consider the laptops example in the formal context in Table 2.1:

Let's take $X = \{L_1, L_2\}$. These two laptops share at least the attributes $\{SSD, Backlit\}$ because both have these attributes. Indeed, $X^{\uparrow} = \{SSD, Backlit\} = Y$. Conversely, $Y^{\downarrow} = \{L_1, L_2\} = X$. Hence, $(X_1, Y_1) = (\{L_1, L_2\}, \{SSD, Backlit\})$ forms a formal concept. This concept pairs a set of objects (the laptops L_1 and L_2) with the exact set of attributes they share (SSD and Backlit). In practice, a formal concept acts as a "fixpoint": no other attributes are missing or superfluous for the chosen set of objects, and no other objects outside X share exactly these attributes.

Moreover, more formal concepts exist, with three represented by the highlighted rectangles in Table 2.2 below:

Table 2.2. Determine Additional Concepts

	T	B	S	D
L_1	×	×	×	
L_2		×	×	
L_3	×		×	×
L_4	×		×	

	T	B	S	D
L_1	×	×	×	
L_2		×	×	
L_3	×		×	×
L_4	×		×	

	T	B	S	D
L_1	×	×	×	
L_2		×	×	
L_3	×		×	×
L_4	×		×	

Specifically, $(X_2, Y_2) = (\{L_1, L_3, L_4\}, \{T, S\})$, $(X_3, Y_3) = (\{L_3\}, \{T, S, D\})$ and $(X_4, Y_4) = (\{L_1, L_2, L_3, L_4\}, \{S\})$.

Beyond individual concepts, FCA also provides a way to arrange them into a hierarchy. Concepts are naturally ordered by a subconcept-superconcept relation, reflecting the intuitive idea that some concepts are more specialized (fewer objects, more attributes) and others more general (more objects, fewer attributes).

DOI: 10.14750/ME.2026.038

Definition 2.4 (Subconcept-Superconcept Ordering) [10]:

For two formal concepts (X_1, Y_1) and (X_2, Y_2) in (G, M, I),

$$(X_1, Y_1) \leq (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2 \Leftrightarrow Y_2 \subseteq Y_1 .$$

If one concept's extent is contained in another's, it is deemed more specific (a subconcept). If we think of real-world categories like "High-End Laptops" being a subconcept of "All Laptops with SSD," this aligns perfectly: "High-End Laptops" is more specialized, fitting strictly inside a larger, more general category. Collecting all formal concepts of a formal context and ordering them by $\leq$ (represents the subconcept-superconcept ordering) yields a concept lattice. This lattice organizes all concepts into a cohesive structure, displaying the entire "map" of how concepts relate in terms of specificity and generality.

Definition 2.5 (Concept Lattice):

In general lattice theory, a lattice is a partially ordered set $(P, \leq)$ in which every pair of elements has a unique greatest lower bound (meet) and a unique least upper bound (join).

In FCA, the collection of all formal concepts derived from a formal context (G, M, I), denoted by

$$\mathcal{B}(G, M, I) = \{(X, Y) \in 2^G \times 2^M \mid X^{\uparrow} = Y, Y^{\downarrow} = X\}.$$

together with the subconcept–superconcept ordering $\leq$, forms a concept lattice. Thus, $(\mathcal{B}(G, M, I), \leq)$ is a lattice where each pair of formal concepts has a unique meet and join, making it the central structural representation in FCA.

In addition to simply being a partially ordered set, a concept lattice is, by definition, a lattice. In general lattice theory, a lattice is a partially ordered set (poset) in which every pair of elements has both a greatest lower bound (meet) and a least upper bound (join). When we say that $(\mathcal{B}(G, M, I), \leq)$ is a lattice, we mean that for any two formal concepts in this set, there is a well-defined concept that serves as their greatest lower bound and another that serves as their least upper bound [10].

- Greatest Lower Bound (Meet or Infimum):

Consider two formal concepts (X_1, Y_1) and (X_2, Y_2) in the concept lattice $\mathcal{B}(G, M, I)$. Their meet, denoted $(X_1, Y_1) \wedge (X_2, Y_2)$, is a formal concept that lies below or equal to both (X_1, Y_1) and (X_2, Y_2) in the ordering $\leq$, and it is the greatest such concept with this property. Intuitively, this meet concept represents the most specific (or "lowest") concept that is still a common "descendant" of both (X_1, Y_1) and (X_2, Y_2). In practical terms, the meet corresponds to the concept formed by taking the intersection of the object sets and determining which attributes remain common to those objects, thus ensuring you get the greatest common "lower" concept.

DOI: 10.14750/ME.2026.038

– Least Upper Bound (Join or Supremum):

Similarly, the join, denoted $(X_1, Y_1) \vee (X_2, Y_2)$, is the least upper bound of the two concepts. It is a formal concept that ranks above or equal to both (X_1, Y_1) and (X_2, Y_2) and is the least such concept with this property. Intuitively, the join concept represents the most general (or "highest") concept that can be seen as a common "ancestor" of both (X_1, Y_1) and (X_2, Y_2). Concretely, you can think of it as taking the union of their attribute sets and finding all objects that share these combined attributes. This ensures you obtain the smallest concept higher than both initial concepts.

Because every pair of concepts in $\mathcal{B}(G, M, I)$ has a unique meet and a unique join, the structure $(\mathcal{B}(G, M, I), \leq)$ qualifies as a lattice.

Definition 2.6 (Extents and Intents):

– The set of all extents of the concept lattice is $Ext(G, M, I) = \{X \subseteq G \mid (X, Y) \in \mathcal{B}(G, M, I), \text{for some } Y\}$.
– The set of all intents is $Int(G, M, I) = \{Y \subseteq M \mid (X, Y) \in \mathcal{B}(G, M, I)\ for\ some\ X\}$.

In summary, starting from a formal context, we derive formal concepts that pair subsets of objects with subsets of attributes, forming a concept lattice when organized by the natural subconcept-superconcept relation. This lattice provides a complete and structurally rich representation of the relationships present in the original data.

Example 2.3 (Extended Laptop Scenario):

To further illustrate the process of deriving a concept lattice, let's consider an expanded example using a set of laptops and additional attributes. Suppose we have the following set of objects (laptops) and attributes:

Objects (Laptops):
$G = \{L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8\}$.

Attributes:
$M = \{T(Touchscreen), B(Backlit), S(SSD), D(Detachable), L(Lightweight), M(Metal\ Chassis), G(Gaming)\}$.

Assume the incidence relation I (indicated by "×") is given by the cross-table in Table 2.3:

Table 2.3. The Extended Laptop Formal Context (Cross-Table Representation)

	T	**B**	**S**	**D**	**L**	**M**	**G**
L_1	X	X	X		X		
L_2		X	X			X	
L_3	X		X	X			X
L_4			X				
L_5	X	X			X	X	
L_6	X			X	X		
L_7			X		X	X	X
L_8	X	X		X	X		X

DOI: 10.14750/ME.2026.038

The corresponding formal context (G, M, I) for our expanded laptop example yields a set of formal concepts. All formal concepts derived from the cross-table described in Table 2.3 are presented in Table A.1 of Appendix A. Each concept provides insight into how certain groups of laptops share defining attributes, ultimately forming the building blocks of the concept lattice.

The corresponding concept lattice $\mathcal{B}(G, M, I)$ is depicted in the following Figure 2.1, as a Hasse diagram. Each node corresponds to one of the formal concepts listed above, and edges illustrate the subconcept-superconcept ordering.

In this dissertation, the set of formal concepts was systematically enumerated using the NextClosure algorithm**,** which guarantees completeness by generating each concept in lectic order. To establish the covering relation of the lattice (i.e., the Hasse diagram edges), the iPred algorithm [16] was applied, as it efficiently predicts direct predecessors without computing all pairwise comparisons. All these steps were implemented in a Python script developed for this research, which automated the extraction of formal contexts, the computation of formal concepts, and the construction of the lattice. Finally, the resulting lattice was visualized through Python libraries such as networkx, and matplotlib, enabling the rendering of the Hasse diagrams.

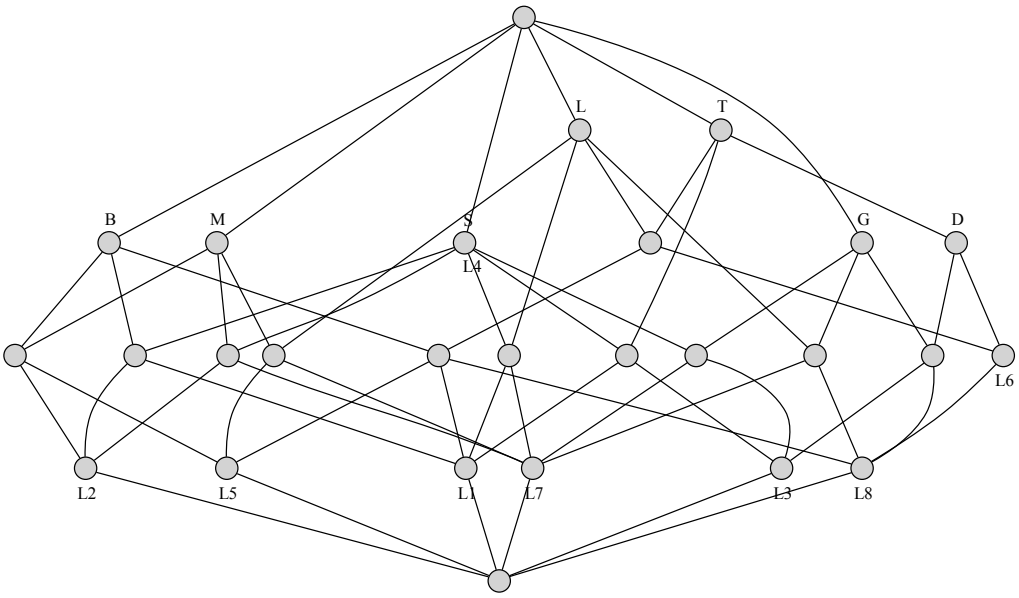


Figure 2.1. Hasse Diagram of the Concept Lattice Derived from the Extended Laptop Context

2.2.4. Hasse Diagram

The Hasse diagram derived from a formal context is a graphical representation that organizes all formal concepts into a hierarchy defined by the subconcept-superconcept relation. Each node in the diagram corresponds to a formal concept, which is composed of an extent (a set of objects) and an intent (a set of attributes). This visual structure allows one to understand how concepts relate to each other, moving from broader, more general concepts near the top, more specific concepts near the bottom.

A standard labeling convention is employed to improve readability and interpretation. Formal concepts that correspond exactly to a single object and all its associated attributes

DOI: 10.14750/ME.2026.038

are called object concepts. For an object $g \in G$, such an object concept is represented as $(\{g\}'', \{g\}')$, and denoted by $\gamma(g)$. In the diagram, the object's identifier g is placed below its corresponding node. As an illustration, if we have an object concept $(\{L_1\}'', \{L_1\}')$ from the laptop formal context above, describing a particular laptop L_1 and its attributes $\{T, B, S, L\}$, the node in the Hasse diagram would be labeled simply as "L_1" beneath it as shown in Figure 2.1, making it immediately clear that this concept is tightly linked to a single object L_1.

Conversely, formal concepts that correspond exactly to a single attribute and all objects sharing it are called attribute concepts. For an attribute $m \in M$, such an attribute concept is represented as $(\{m\}', \{m\}'')$ and denoted by $\mu(m)$. The attribute's identifier m is placed above its corresponding node. For example, if there is an attribute concept ($\{\{L_1, L_2, L_5, L_8\}, \{B\}$) focusing on all laptops that have the attribute B, the node would be labeled simply "B" above it as shown in Figure 2.1, signaling that this concept captures the essence of attribute B.

This labeling approach, objects below and attributes above is not only consistent but highly informative. It allows for quick identification of object concepts and attribute concepts. At a glance, one can recognize pure object or attribute concepts by their placement of labels. However, not every concept fits neatly into these two categories. Many formal concepts arise from more complex intersections of multiple objects and attributes. Such concepts may represent meaningful clusters or patterns in the data without corresponding purely to one object or one attribute. These intermediate nodes often remain unlabeled or require more careful interpretation of their extents and intents [17], [18].

By traversing the diagram from bottom to top, one moves from general concepts (which may have many objects and fewer attributes) to more specialized concepts (fewer objects, more attributes). The topmost node often represents a concept characterized by a maximal set of attributes (sometimes all attributes), and possibly no objects at all, while the bottommost node often represents a concept containing all objects and possibly no attributes. Intermediate nodes show how objects group together under shared attribute sets and how attribute sets apply to particular object subsets.

In summary, the Hasse diagram's layout and labeling conventions provide a clear, intuitive framework for interpreting complex data relationships. Object labels below nodes, attribute labels above nodes, and unlabeled intersections work together to reveal how objects and attributes interact, cluster, and form meaningful concepts. This makes the concept lattice an invaluable tool for gaining insights, identifying patterns, and informing decision-making in a wide range of application domains.

2.2.5. Properties of FCA

Let (G, M, I) be a formal context, with G as a non-empty set of objects, M as a non-empty set of attributes, and $I \subseteq G \times M$ denoting the incidence relation. Consider arbitrary subsets $X, X_1, X_2 \subseteq G$ and $Y, Y_1, Y_2 \subseteq M$. Recall the concept-forming operators: for $X \subseteq G$, define

$$X^{\uparrow} = \{m \in M \mid \forall g \in X, (g, m) \in I\},$$

and for $Y \subseteq M$, define,

$$Y^{\downarrow} = \{g \in G \mid \forall m \in Y, (g, m) \in I\}.$$

DOI: 10.14750/ME.2026.038

These operators $(\cdot)'$ map subsets of objects to subsets of attributes and vice versa. From the definitions of $(\cdot)'$ and related properties, the following fundamental relationships hold [10]:

- Monotonicity of Derivations (for Objects):
  If $X_1 \subseteq X_2 \subseteq G$, then

$$X_1{}' \supseteq X_2{}'.$$

In other words, enlarging the set of objects cannot produce a larger attribute set. Instead, it can only stay the same size or become smaller.

- Monotonicity of Derivations (for Attributes):
  If $Y_1 \subseteq Y_2 \subseteq M$, then

$$Y_1{}' \supseteq Y_2{}'.$$

Analogously, increasing the set of attributes leads to a smaller or equal set of objects sharing all of them.

- Closure-like Properties of Double Derivations:
  For any $X \subseteq G$, we have

$$X \subseteq X'' and\ X' = X'''.$$

Similarly, for any $Y \subseteq M$,

$$Y \subseteq Y'' and\ Y' = Y'''.$$

Here, $X'' = (X')'$ and $X''' = (X'')'$ are iterated derivations. The equality $X' = X'''$ and $Y' = Y'''$ indicate a closure-like behavior of these operators.

- Galois Connection-Style Equivalences:
  The up and down derivations $(\cdot)'$ satisfy a Galois connection between subsets of G and subsets of M. Specifically,

$$X \subseteq Y' \iff X' \subseteq Y.$$

This property encapsulates the fundamental duality: a set of objects X is included in the down-set of a set of attributes Y if and only if the attribute set X' is included in Y.

Formal Concepts as Fixpoints: A pair (X, Y) with $X \subseteq G$ and $Y \subseteq M$ forms a formal concept if and only if

$$X' = Y \text{ and } Y' = X.$$

This fixpoint condition ensures that (X, Y) captures a perfectly correlated cluster of objects and attributes, no object outside X shares all attributes of Y, and no attribute outside Y is common to all objects in X. Formal concept analysis encompasses a broad range of properties and advanced structures. For more intricate details, mathematical proofs, and

DOI: 10.14750/ME.2026.038

generalizations, authoritative references include [11], which provides a comprehensive mathematical foundation, as well as Carpineto and Romano [19], among others.

2.2.6. Central Theorem on Concept Lattices

Before stating the main theorem, we briefly recall some notions from lattice theory. A complete lattice $(L, \leq)$ is a lattice in which every subset of L has both a supremum (least upper bound, denoted $\bigvee$) and an infimum (greatest lower bound, denoted $\bigwedge$). This property ensures that arbitrary joins and meets exist, not only those for finite pairs. Within such a structure, certain subsets play a special role: a subset $S \subseteq L$ is called supremum-dense if every element of L can be expressed as the join (supremum) of some subset of S. Likewise, a subset $T \subseteq L$ is called infimum-dense if every element of L can be expressed as the meet (infimum) of some subset of T.

The main theorem by Wille (1982) provides a fundamental characterization of concept lattices [1]:

Theorem (Main Theorem of Concept Lattices):

Let (G, M, I) be a formal context and $\mathcal{B}(G, M, I)$ its set of formal concepts. Then:

1. Completeness:
   $(\mathcal{B}(G, M, I), \leq)$ is a complete lattice. For any collection of formal concepts $\{(X_j, Y_j)\}_{j \in J}$:

   $$\bigwedge_{j \in J}(X_j, Y_j) = \left(\bigcap_{j \in J} X_j, \left(\bigcup_{j \in J} Y_j\right)''\right), \ \bigvee_{j \in J}(X_j, Y_j) = \left(\left(\bigcup_{j \in J} X_j\right)'', \bigcap_{j \in J} Y_j\right).$$

2. Representation of Complete Lattices:
   Any complete lattice $(V, \leq)$ can be represented as a concept lattice $\mathcal{B}(G, M, I)$ if there exist mappings $\gamma: G \to V$ and $\mu: M \to V$ such that:
   - $\gamma(G)$ is supremum-dense ($\bigvee -dense$) in V and $\mu(M)$ is infimum-dense ($\bigwedge -$dense) in V.
   - $\gamma(g) \leq \mu(m)$ if and only if $(g, m) \in I$.

The theorem ensures $\mathcal{B}(G, M, I)$ is always a complete lattice, with well-defined supremum and infimum operations. It also shows that concept lattices are universal: any complete lattice can be modeled as a concept lattice by choosing appropriate objects and attributes. Labeling the lattice by assigning each object g to its object concept $\gamma(g)$ and each attribute m to its attribute concept $\mu(m)$ preserves all original information. This result anchors concept lattices as a core mathematical structure in FCA, ensuring no loss of information and offering a canonical form for representing complete lattices.

2.3. Overview of FCA algorithms

Computing all formal concepts from a given formal context can be challenging. A naive approach will examine every subset of attributes $Y \subseteq M$ to determine if it forms a closed set.

DOI: 10.14750/ME.2026.038

This would be computationally prohibitive since there are $2^{|M|}$ subsets of attributes. Instead, FCA algorithms incorporate strategies like canonical test conditions, lexicographic ordering of subsets, and efficient data structures to prune the search space. FCA encompasses a variety of algorithms designed to efficiently construct concept lattices from formal contexts. These algorithms can be broadly categorized into three classes: Batch-Style Computation, Incremental Techniques for Update, and Assembling Algorithms. Each class employs distinct methodologies tailored to specific computational and application requirements. This section provides an exploration of these algorithmic classes, highlighting their operational principles [20].

2.3.1. Batch-Style Computation

Batch-style algorithms are foundational in FCA, generating the entire set of formal concepts from scratch by processing the complete formal context in a single run. These algorithms typically operate in a top-down or bottom-up manner, either beginning with minimal intents and progressively building towards maximal intents or vice versa. A notable technique within this category is the lexicographic ordering of attribute subsets, which serves to streamline the concept generation process by preventing redundant computations.

One of the most prominent batch algorithms is Ganter's NextClosure algorithm, introduced in [10], [21]. The algorithm employs a lexicographic order to systematically explore subsets of attributes, ensuring that each formal concept is generated exactly once in a canonical form. The key idea is to traverse the space of attribute subsets in a predetermined order, applying closure operations to identify and confirm formal concepts. The following pseudo-code of the NextClosure algorithm is adapted from [10].

Algorithm 2.1: NextClosure

Input:
A formal context (G, M, I), with attribute set $M = \{m_1, m_2, \dots, m_n\}$.
A current intent $Y \subseteq M$.
Output:
All formal concepts derived from Y, enumerated in lectic (canonical) order.

```
procedure NextClosure(Y):
   output concept (Y', Y)              // extent-intent pair
   i ← n
   success ← false
   while not success and i > 0 do
      i ← i − 1
      if m_i ∉ Y then
         D ← Y ∪ {m_i}
         C ← (D)''                     // closure of D
         if (C \ Y) contains no element < m_i then
            NextClosure(C)             // recursive call
            success ← true
         end if
      end if
   end while
end procedure
```

NextClosure operates with a time complexity of $O(|\ G\ |^2 \cdot |\ M\ | \cdot |\ L\ |)$, where $|\ L\ |$ is the number of formal concepts. Its polynomial delay of $O(|\ G\ |^2 \cdot |\ M\ |)$ makes it efficient for contexts with moderate sizes.

DOI: 10.14750/ME.2026.038

Another significant batch algorithm is the Close-by-One (CbO) family, which enhances the concept generation process by incorporating efficient canonicity tests and leveraging data structures to store intermediate results. The original CbO algorithm, detailed in [22], focuses on reducing redundant computations through strategic exploration of the attribute space. The Fast Close-by-One (FcbO) algorithm, introduced in [23], enhances CbO by implementing an additional canonicity test to eliminate redundant computations. FcbO employs a breadth-first search strategy to propagate canonicity failures, thereby significantly reducing the number of attribute comparisons required. Another advanced variant is the In-Close algorithm [24], which optimizes closure operations by incrementally computing closures and employing matrix searching techniques. The In-Close2 version [25] further refines this approach by propagating attribute tests downward and reordering the context table to group maximal rectangles, resulting in performance gains over FcbO.

Batch algorithms like NextClosure and CbO are powerful for generating complete concept lattices from static contexts. However, their performance can degrade with large datasets due to the exponential number of possible attribute subsets. Despite optimizations, these algorithms require re-computation from scratch whenever the formal context changes, limiting their applicability in dynamic environments.

2.3.2. Incremental Techniques for Update

Incremental algorithms address the limitations of batch algorithms by efficiently updating the concept lattice in response to changes in the formal context, such as the addition or removal of objects or attributes. These algorithms build the concept lattice incrementally, modifying the existing structure with minimal computational overhead rather than reconstructing it entirely.

One of the earliest incremental algorithms is Norris' Algorithm [26], which updates the concept lattice by sequentially incorporating new objects. The algorithm maintains the lattice structure by identifying and adjusting affected concepts when a new object is introduced. AddIntent [27] is another algorithm enhances incremental updates by employing heuristics to identify modified and generator concepts. It traverses the concept lattice graph recursively, ensuring that each new concept is processed exactly once and that canonicity is maintained throughout the update process. Another approach is presented in [28], where an optimized incremental concept lattice construction method improves update efficiency by integrating features from Ferre and InClose algorithms. This approach reduces computational complexity through context reduction techniques, enabling faster concept set updates. The method outperforms existing incremental techniques in specific parameter ranges, making it particularly effective for dynamic datasets that require frequent modifications.

Further advancements in this field have introduced more sophisticated strategies to address scalability, modularity, and adaptability in dynamic environments. One such advancement focuses on the incremental mining of frequent closed itemsets (FCIs), where the aim is to avoid full recomputation when transactions or minimum support thresholds change. This approach, presented in [29], outlines two strategies, one mirroring classical lattice traversal and another optimized via structural pruning that significantly reduce redundant computation by localizing updates to affected sublattices. The method proves particularly effective for dynamic databases and exploratory workflows, where the frequent tuning of parameters like minsup is common.

DOI: 10.14750/ME.2026.038

In a broader lattice-assembly perspective, [30] introduces a general framework for incremental object insertion that classifies existing lattice concepts into four distinct categories: new, modified, generator, and unchanged. Each category undergoes localized transformations that preserve the global lattice structure with minimal disruption. This classification not only guides efficient lattice maintenance but also connects incremental updates with modular lattice construction through the Compute-Lattice-Inc procedure, which incrementally builds a global lattice by composing partial ones.

Expanding on this foundation, [31] proposes a generic insertion scheme based on structural invariants that constrain how upper and lower covers change during incremental updates. A key result here is the formulation of the ADD-OBJECT algorithm, which scans concept intents against new object attributes, augments extents as needed and ensures correctness via targeted ORDER updates. Beyond single insertions, this framework extends to context subposition and distributed assembly, providing robust support for batch updates and modular factorization across multiple datasets.

Complementing these approaches, [32] introduces a partition-based construction strategy that embraces a divide-and-conquer philosophy. By fragmenting the input context and computing individual lattices for each fragment, the method reassembles a global lattice through a structured traversal of the product order. It employs specialized data structures such as concept records, ranked lists, and embedding tables and culminates in the ASSEMBLY procedure, which merges partial lattices efficiently. This technique offers a scalable solution for large or naturally partitioned datasets and conceptually bridges incremental and modular construction paradigms.

Incremental algorithms provide a robust solution for maintaining concept lattices in dynamic environments. By updating the lattice incrementally, these algorithms offer substantial performance improvements over batch algorithms, especially in scenarios with frequent data modifications. Their ability to efficiently handle updates makes them indispensable for applications involving real-time data streams and evolving datasets.

2.3.3. Assembling Algorithms

Assembling algorithms represent an evolution of incremental techniques, focusing on constructing concept lattices by combining partial structures derived from segmented parts of the formal context. This approach is particularly advantageous for large-scale and distributed datasets, as it allows for parallel processing and efficient aggregation of results.

The Divide & Conquer algorithm [33] exemplifies this class by partitioning the formal context into smaller, manageable segments, computing concept lattices for each segment independently, and subsequently merging these partial lattices into a cohesive whole. This method leverages parallelism, enabling significant scalability and efficiency gains by distributing the computational load across multiple processors or machines.

Key Steps of Assembling Algorithms:

1. Partitioning the Formal Context:
   - The formal context (G, M, I) is divided either vertically (by attributes) or horizontally (by objects) into smaller subcontexts.
2. Computing Partial Lattices:
   - For each subcontext, a partial concept lattice is constructed using batch or incremental algorithms.
3. Merging Partial Lattices:

DOI: 10.14750/ME.2026.038

- The partial lattices are then combined, ensuring consistency and maintaining the overall lattice structure. This often involves resolving overlaps and integrating shared concepts.

Advantages of Assembling Algorithms:

- Parallel Processing: Enables concurrent computations on different segments of the formal context, significantly reducing total computation time.
- Scalability: Efficiently handles large datasets by distributing the processing workload, making it suitable for big data applications.
- Flexibility: Can be adapted to various partitioning strategies, allowing optimization based on specific dataset characteristics and computational resources.

Assembling algorithms extend the capabilities of incremental and batch methods by facilitating the construction of concept lattices from segmented data. Their inherent parallelism and scalability make them particularly suited for handling extensive and complex datasets, ensuring that FCA remains effective even as data volumes grow.

2.3.4. General Remarks on FCA Algorithm's Performance

The computational efficiency of FCA algorithms is influenced by several parameters associated with the formal context (G, M, I). Key factors include:

- The cardinalities $|G|$ and $|M|$.
- The size of the incidence relation I.
- The density $\rho = \frac{|I|}{|G| \,.\, |M|}$, which measures how densely attributes occur in objects.

Empirical evaluations [20] indicate that certain algorithms excel under specific conditions. For contexts where $|G|$ and $|M|$ are large and ρ is high (dense contexts), bottom-up algorithms like Close-by-One and NextClosure, as well as Norris' incremental algorithm, often yield superior performance. Conversely, for contexts where $|G|$ and $|M|$ remain small and ρ is low (sparse contexts), incremental methods such as Godin's algorithm[34] can be more effective, thereby minimizing unnecessary computations.

Constructing the entire lattice, including its partial order, imposes an additional computational burden. Algorithms solely generating the set of formal concepts C often exhibit lower runtime complexity than those that must also determine the ordering relations $\leq$ among concepts. This is because the calculation of minimal upper neighbors and lower neighbors for each concept in C introduces extra steps beyond the initial concept generation.

Preventing redundant concept computations is essential for efficiency. The set of all formal concepts derived from (G, M, I). Without careful checks, an algorithm might attempt to recompute concepts it has already enumerated. To address this, suitable data structures and canonicity checks are employed. For instance, Godin's algorithm organizes concepts by the cardinality of their intents, enabling quick lookups and pruning strategies.

A commonly used technique to accelerate set operations involves representing attribute subsets as fixed-length bit arrays. Each attribute $m \in M$ corresponds to a particular bit position. For two subsets $X, Y \subseteq M$, the set intersection $X \cap Y$ translates directly to a bitwise AND operation on their corresponding bit arrays. This representation reduces set-theoretic operations to $O(|\ M\ |/w)$ time, where w is the word size of the machine. For example, if

DOI: 10.14750/ME.2026.038

$M = \{a < b < c < d < e < f\}$ and $S = \{a, c, d, e\}$, one can encode S as a binary vector $v(S) = 101110$, where the position of each bit corresponds to an attribute in lexicographic order. Then, $X \cap Y$ is computed as $v(X) \wedge v(Y)$, a single bitwise operation.

Additionally, reordering rows and columns of the context (G, M, I) can yield substantial improvements. Sorting attributes by frequency and rearranging objects to minimize Hamming distances between their bit-array representations enhances spatial locality and cache utilization. Such techniques, as implemented in In-Close2 [25], demonstrate performance gains in excess of 30% for large datasets.

In conclusion, the performance of FCA algorithms depends not only on the choice of algorithmic strategy, bottom-up vs. incremental, but also on data representation techniques, the selection and design of indexing structures, and preprocessing steps like ordering and clustering of attributes and objects. By judiciously combining these strategies, FCA computations can scale more efficiently to handle increasingly large and dense formal contexts.

2.4. Extensions and Applications of FCA Model

This section explores various extensions and applications of the FCA model, highlighting advancements that enhance both its theoretical foundations and practical utility. One significant extension is the concept of box elements in a concept lattice, introduced as a refinement of FCA [35]. This work focuses on constructing the box lattice of a given concept lattice $\mathcal{B}(G, M, I)$, which serves as a structured framework for classification systems. The box lattice, derived from a CJ-generated complete lattice (Completely Join-irreducible generated), allows for an alternative decomposition of concept lattices into atomistic components, enhancing the classification hierarchy's representation. The study establishes the equivalence between classification lattices and box lattices, proving that any classification lattice can be reconstructed from an atomistic complete lattice. Additionally, the paper proposes an algorithmic approach for computing box elements, which is particularly useful in cluster analysis and group technology applications. This extension of FCA provides a novel method for structuring and analyzing classification systems within concept lattices.

Building upon this, an important computational enhancement of FCA is presented in [36], where the authors introduce an incremental method for constructing box extents in a concept lattice. This research improves the efficiency of box element construction by refining the one-object extension method, demonstrating that box extents can be incrementally generated while avoiding exponential growth in complexity. A key result is that the box extent lattice can be order-embedded in the lattice of atomic extents, further strengthening FCA's mathematical foundation for classification tasks. The paper contributes a computationally feasible algorithm that improves the practical applicability of FCA in data classification, clustering, and engineering applications, particularly in Group Technology. By optimizing the successive extension of box extents, this work significantly enhances the scalability and usability of FCA-based classification systems.

Beyond the crisp (binary) framework of traditional FCA, another key extension into the fuzzy domain is presented in [37]. This work explores the lattice structure of fuzzy rough sets with crisp reference sets, integrating fuzzy logic with FCA principles to handle graded membership and uncertainty in classification. The study establishes an isomorphism between the lattice of fuzzy rough sets and the lattice of rough sets for a crisp equivalence relation, preserving fundamental order-theoretic properties crucial to FCA. By bridging

DOI: 10.14750/ME.2026.038

fuzzy rough set theory with concept lattice structures, this extension allows FCA to be applied in uncertain knowledge representation, particularly in machine learning, data classification, and artificial intelligence. This adaptation expands the scope of FCA beyond traditional binary relations, enabling its use in complex, real-world scenarios where uncertainty must be considered.

FCA's diverse applications extend beyond computer science to statistics, medicine, psychology, social sciences [12], [13], [14], artificial intelligence, and information retrieval [9]. In education, FCA is applied to student assessment by constructing concept lattices from a Student Assessment Matrix (SAM) and a Question Skill Matrix (QSM) [38]. This approach visualizes student knowledge hierarchies, enabling objective grading, personalized learning, and effective group formation based on complementary skills. By integrating FCA into educational evaluation, this method offers a data-driven framework for assessing student performance and knowledge representation. Beyond education, FCA has been effectively applied in industrial engineering [39], particularly in solving the machine-part grouping problem in cellular manufacturing. By analyzing a binary incidence matrix, formal concepts and extent partitions are used to optimize manufacturing cells, improving machine utilization and reducing inter-cell movements, this application further demonstrates FCA's versatility in addressing complex, real-world industrial challenges. In Natural Language Processing (NLP), FCA is applied to part-of-speech (POS) classification by constructing concept lattices to generate classification rules [40]. This FCA-based approach replaces traditional decision trees and neural networks, identifying maximal consistent nodes to improve classification accuracy and efficiency. Experimental results show that the FCA-based classifier outperforms neural networks in execution time and accuracy, making it a viable alternative for morphological classification and linguistic analysis. A closely related application in string transformation rule induction uses FCA to generalize morphological transformations by constructing concept lattices [40]. This approach improves rule induction efficiency by organizing transformation rules hierarchically, enabling compact and generalized rule extraction. The optimized incremental concept lattice construction enhances pattern recognition and linguistic processing, making FCA a valuable tool in text analysis and NLP.

These extensions and applications highlight FCA's adaptability and effectiveness in diverse fields, demonstrating its potential for advancing both theoretical research and practical problem-solving across multiple domains.

2.5. Emerging Issues in FCA and the Necessity for Reduction Methods

FCA's relevance in complex data analysis is tempered by practical constraints, particularly as datasets grow in size, complexity, and heterogeneity. The following subsections highlight key challenges that underscore why reduction techniques are integral to the next generation of FCA methodologies. These insights reflect the ongoing dialogue in the literature regarding computational bottlenecks, data transformation strategies, and maintaining representational fidelity [20], [25].

DOI: 10.14750/ME.2026.038

2.5.1. High-Dimensional and Complex Datasets

As datasets expand in both volume and complexity, the practical application of FCA faces intensified challenges. The sheer scale of modern data potentially encompassing hundreds of thousands of objects and attributes drives exponential growth in the number of formal concepts and the resultant concept lattice [41]. In essence, each new object or attribute can significantly multiply the possible combinations of object-attribute pairs, creating a combinatorial explosion that places tremendous computational burdens on lattice construction and subsequent analysis.

This surge in complexity is particularly evident in domains where data is inherently large-scale and intricate. For example, in bioinformatics and genomics research, ever-increasing datasets contain vast numbers of genes or proteins, each associated with a myriad of attributes such as functional annotations, experimental conditions, and genomic variants [42], [43]. Similarly, space telemetry data, streaming in real time from numerous sensors, and massive e-learning repositories, recording the activities and proficiencies of thousands of learners, yield datasets too extensive for conventional FCA techniques to handle efficiently. Without suitable reduction strategies, the concept lattice may become prohibitively large and overwhelming, making it difficult for analysts to extract meaningful patterns or insights.

Moreover, high-dimensional data often includes attributes of varying scales, types, and significance. Some attributes may be redundant or represent fine-grained distinctions that, while statistically present, hold limited analytical value. Others may be essential but obscured by a plethora of less relevant details. The presence of these “noisy” or low-impact attributes further exacerbates the complexity by producing a multitude of extra concepts that are not necessarily relevant for the task at hand [44]

In response to these challenges, FCA must be equipped with sophisticated reduction techniques designed to operate at scale. For instance, heuristic filtering can prune attributes or objects that fall below a certain frequency threshold, ensuring that only the most prominent and impactful elements remain [30]. Clustering-based approaches can collapse sets of similar rows or columns to simplify the formal context, thereby yielding a more compact and cognitively manageable lattice structure [45], [46], [47]. Additionally, computational methods like Singular Value Decomposition (SVD) or Non-negative Matrix Factorization can compress the data matrix into a reduced representation, though these factorization-based approaches must be applied carefully to preserve interpretability and handle noise gracefully [48].

Ultimately, handling high-dimensional and complex datasets requires a balanced synergy between computational optimization, conceptual abstraction, and selective filtering. By integrating robust reduction methods, FCA can maintain its core strength as a formal, structured approach to understanding complex data relationships, even in environments characterized by massive scale and multifaceted data attributes.

2.5.2. Adapting to Varied Data Forms Through Scaling

FCA traditionally requires binary input data, but many real-world problems involve continuous, many-valued, or more complex data types [49]. Scaling bridges this gap by transforming non-binary attributes into a suitable form. However, determining effective scaling parameters and intervals, particularly in dynamic data scenarios like data streams remains non-trivial [25]. As new data arrives or attribute ranges shift, scaling must be

DOI: 10.14750/ME.2026.038

repeated or updated, further straining computational resources. Efficient incremental or distributed scaling approaches become essential for maintaining performance and achieving timely analysis results.

2.5.3. Handling Uncertainty: Noise and Missing Values

Real-world datasets often contain outliers, incorrect measurements, or incomplete information. Such imperfections can inflate the number of formal concepts, as even slight deviations generate additional, and frequently irrelevant concepts [50]. Mitigating these effects calls for strategies that tolerate a controlled level of exceptions fault-tolerant FCA methods or that apply smoothing, filtering, or imputation techniques to ensure the resulting lattice remains both accurate and concise [44]

Beyond these fundamental issues, FCA grapples with other complexities as data volumes grow:

- Parallel and Distributed Computation: Efficiently computing concept lattices in parallel or distributed environments is crucial as datasets become too large for single-node solutions [51]. Distributed algorithms and load-balancing strategies can significantly improve scalability.
- Data Stream Processing: The high velocity of streaming data demands incremental updating and approximation methods to keep up with new objects and attributes without recomputing entire lattices [52].
- Visualization of Large Lattices: As concept lattices grow large, traditional Hasse diagrams become visually overwhelming and hinder user comprehension. While tools like ConExp [53], Galicia [54] provide different visualization strategies, they still face challenges in interactivity and scalability for large-scale contexts. Figure 2.2 illustrates some default visualization strategies, demonstrating that straightforward approaches are insufficient for extensive datasets.

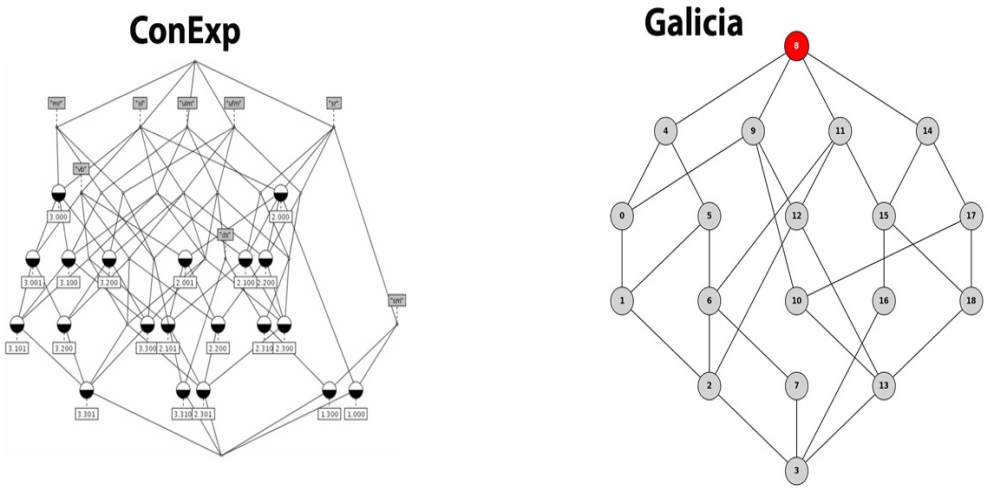


Figure 2.2. Representative Tools for Concept Lattice Visualization

In conclusion, as FCA encounters increasingly complex, large-scale, and evolving datasets, it must integrate refined reduction methods, advanced scaling strategies, and robust mechanisms for noise handling and concept filtering. These enhancements will ensure that

DOI: 10.14750/ME.2026.038

FCA remains both a powerful and practical framework for knowledge discovery in challenging data-driven environments.

2.6. Summary

This chapter provided a comprehensive foundation for FCA, detailing the theoretical underpinnings that make it a powerful framework for knowledge representation. After introducing the essential constructs, it examined how FCA identifies conceptual structures within data and organizes them into concept lattices.

Subsequently, the chapter reviewed key algorithmic paradigms, batch computation, incremental updates, and lattice assembly, emphasizing how each approach addresses different computational challenges. Techniques to enhance algorithmic performance, such as efficient data structures, attribute reordering, and heuristic filtering, were highlighted as essential tools for managing complexity.

The chapter then focused on the pressing challenges FCA faces in contemporary big data contexts, where high-dimensional, noisy, and evolving datasets demand more scalable and flexible solutions. This necessitates robust reduction methods, advanced scaling techniques for complex data types, and approaches to handle noise and missing values. Recognizing these obstacles sets the stage for developing refined methodologies that preserve FCA's conceptual clarity and interpretability, even as data grows in scale and complexity.

DOI: 10.14750/ME.2026.038

Chapter 3: Literature Review

3.1. Overview of Existing Lattice Reduction Techniques in FCA

FCA has, since its conception by Wille in 1982 [1], evolved into a powerful tool for knowledge extraction and structural data analysis. Its mathematical foundations enable the organization of data into concept lattices, revealing intricate relationships that have proved beneficial across a wide range of domains, data mining [6], neural networks [55], and social network analysis [12], [13], [14], among others. Yet, as FCA applications have expanded, the complexity of the resulting lattices has become a significant concern. Reducing the complexity of concept lattices derived from FCA is a critical challenge in knowledge engineering. As FCA extracts formal concepts and arranges them into a concept lattice, the resulting structure can become prohibitively large and intricate, often rendering it computationally demanding and difficult for humans to interpret. Therefore, the quest for effective reduction techniques, approaches that simplify these lattices while retaining their essential conceptual relationshipsm is at the forefront of FCA research.

Various reduction methodologies have been proposed to manage and simplify concept lattices [56]. They generally fall into three categories: redundancy removal, simplification, and selection-based strategies. Redundancy removal methods focus on eliminating unnecessary objects, attributes, or incidences, ensuring the resulting lattice remains isomorphic to the original but reduced in scale [57], [58], [59]. However, while these methods can yield smaller, structurally similar lattices, they often remain computationally heavy for very large datasets and do not always provide a significantly more interpretable structure. A straightforward reduction method in this category involves merging multiple objects that share exactly the same attributes into a single representative object or merging multiple attributes that appear together across identical sets of objects into one attribute. By removing these redundancies, the resulting formal context becomes “clarified” while retaining the same conceptual structure [10]. Another type of reduction that maintains the lattice’s overall structure is to remove any attribute that can be represented by other existing attributes, referred to as a reducible attribute[10]. Formally, if there is an attribute $m \in M$ and a subset of attributes $B \subseteq M$, with $m \notin B$, such that $m' = B'$, then μm (the attribute concept of m) is the infimum of the attribute concepts $\mu(b)$ for all $b \in B$. Consequently, if attribute m is removed, the resulting concept lattice remains equivalent to the original one, both in structure and relational ordering. In a similar way, eliminating reducible objects from a formal context can yield a smaller context whose associated concept lattice is still isomorphic to that of the original. Specifically, an object $g \in G$ for which $\gamma(g)$ is the infimum of $\gamma(a)$ over some set $A \subseteq G$ and $g \notin A, a \in A$,can be removed without changing the isomorphism class of the resulting [10]. To further minimize the size of formal contexts while preserving the underlying concept lattice, various strategies have been developed. One such strategy, proposed in [60], uses a “discernibility matrix” to determine a minimal subset of attributes. This approach treats (G, M, I) as a formal context and looks at pairs of concepts $(A_1, B_1), (A_2, B_2) \in \mathcal{B}(G, M, I)$. The symmetric difference of their intention parts, $= (B_1 \cup B_2) \backslash (B_1 \cap B_2)$, defines their “discernibility.” Once the discernibility matrix is constructed, a minimal set of attributes $B \subseteq M$ can be chosen so that the resulting lattice $\mathcal{B}(G, M, I')$ remains isomorphic to $\mathcal{B}(G, M, I)$. Here, $I' = I \cap (G \times A)$, and A denotes the minimal set of attributes having the smallest cardinality. Building on this, Qi [61] presented guidelines to

DOI: 10.14750/ME.2026.038

reduce the number of discernibility computations, still ensuring the possibility of obtaining a minimal set of attributes. Furthermore, [60] categorize attributes of a formal context as "absolutely necessary," "relatively necessary," or "absolutely unnecessary." An attribute that appears in every minimal set is deemed absolutely necessary; if it appears in at least one but not all minimal sets, it is relatively necessary; and if an attribute appears in none of the minimal sets, it is considered absolutely unnecessary.

Simplification or abstraction approaches attempt to approximate or restructure the lattice to emphasize its most essential features while accepting some information loss. These include clustering similar objects or attributes to form more compact representations [62], employing algebraic reductions like SVD or non-negative matrix factorization [63], and leveraging approximation operators such as neighborhood-based concept lattices [64]. The discernibility matrix-based reduction algorithm [65] and sophisticated factorization methods [66] represent key efforts to minimize complexity. Linguistic-valued layered lattice simplifications that consider three-way decision methods [67], as well as attribute reduction in Pythagorean Fuzzy formal contexts leveraging optimized Apriori-algorithm variants [68], reflect ongoing innovation in this area. Yet, these simplification methods may rely on assumptions (e.g., pseudo similarities) or introduce computational burdens that limit their practical use. An interesting approach related to focusing on a sublattice of concept lattices rather than enumerating all concepts, was presented in [35]. They consider row-reduced contexts and define the so-called *box lattice*, $Box(B(G,M,I))$, which retains exactly those concepts relevant for classification systems, leading to an atomistic sublattice. Their method identifies and generates these 'box elements' by finding the atoms of $Box(B(G,M,I))$. This atom-based decomposition provides a systematic framework to study or build classification lattices in a potentially more manageable subset, thereby extending formal concept analysis techniques to clustering and grouping tasks where classification systems play a key role.

Selection-based techniques provide another promising avenue for lattice reduction. Instead of attempting to maintain or approximate the entire structure, selection strategies isolate only those concepts or attributes deemed most relevant for a given analysis [18], [56]. A particularly influential reduction strategy is the Iceberg Lattice, introduced in [43]. Unlike algebraic or attribute-based simplifications, this approach offers a scalable solution for reducing the size of concept lattices in FCA. Its central principle is to retain only the "top-most" portion of the lattice by filtering concepts according to a minimum support threshold. In doing so, the method preserves the most frequent and globally significant concepts, while pruning less frequent and potentially less informative ones.

Formal Definition:

Let $K = (G, M, I)$ be a formal context, where G is the set of objects, M is the set of attributes, and $I \subseteq G \times M$ is the incidence relation.

For an attribute set $B \subseteq M$, its support is defined as:

$$supp(B) = | B' | / | G |.$$

where $B' \subseteq G$ is the set of all objects possessing attributes in B.

An attribute set B (or concept intent) is called frequent if:

$$supp(B) \geq minsup, \text{ with } minsup \in [0,1].$$

DOI: 10.14750/ME.2026.038

- A frequent concept is a formal concept (A, B) whose intent B is frequent.
- The collection of all frequent concepts forms the iceberg concept lattice of K.

Since support is monotone decreasing with set inclusion (i.e., if $B_1 \subseteq B_2$ then $supp(B_1) \geq supp(B_2)$, the iceberg lattice is an order filter of the full lattice. In practice, this means it generally forms only a join-semilattice. To restore lattice completeness, a new bottom element can be artificially introduced. The Authors proposed the TITANIC algorithm (a *level-wise* data mining procedure) to efficiently compute iceberg concept lattices. The key innovation lies in introducing a weight function in this case, the support function that is compatible with the FCA closure operator. This compatibility enables the algorithm to prune the search space and avoid redundant computations.

The algorithm proceeds as follows:

1. Initialization: Begin with the empty set and all singleton attribute sets as candidate generators.
2. Weight Calculation: Compute support values for candidate sets.
3. Closure Computation: Determine the closure of candidate sets by checking support invariance under attribute addition.
4. Key Sets Identification: Identify minimal generators (key sets) whose closures yield new frequent concepts.
5. Pruning: Discard non-key sets or those below the support threshold.
6. Iteration: Generate higher-level candidates by combining frequent subsets.
7. Termination: Stop when no new frequent sets remain.

This level-wise exploration ensures that only frequent concepts above the support threshold are considered, dramatically improving scalability compared to classical approaches like Ganter's NextClosure algorithm. In their seminal work, Stumme et al. applied iceberg lattices to the MUSHROOM database from the UCI repository.

- The full lattice contained 32,086 concepts, far too large for practical visualization.
- By applying a minimum support of 85%, the iceberg lattice was reduced to only the most frequent concepts, highlighting attributes such as veil type: partial (100% support), veil color: white (97.62%), and gill attachment: free (97.43%).
- Decreasing the support threshold to 70% or 55% progressively revealed finer structures and associations, including implications like.

$$\{gill\ attachment\colon\ free,\ gill\ spacing\colon\ close\} \Rightarrow \{veil\ color\colon\ white\}.$$

Thus, iceberg lattices act as a multi-resolution tool, where lowering the support threshold became apparent. In this way, iceberg lattices can be viewed as a multi-resolution tool: higher thresholds expose the most dominant and general patterns, while lower thresholds allow exploration of finer and more specific associations.

Beyond their role in lattice reduction, iceberg lattices have become a central tool in related areas of knowledge discovery. They provide a condensed representation of frequent closed itemset, support efficient association rule mining through non-redundant bases, and enable structured visualization of large datasets that would otherwise be computationally prohibitive and cognitively overwhelming. By systematically filtering concepts based on

support, iceberg lattices strike a balance between scalability and interpretability, laying the groundwork for many subsequent advances in FCA-based reduction and mining techniques. While this approach offers a straightforward means of reducing lattice size and highlights frequently occurring concepts, it does not account for derivation costs; concepts with the same high support remain equally in the lattice, even if one is far more central for deriving other concepts. By narrowing down the concept set, these methods can achieve more manageable and interpretable lattices.

In many scenarios, there is additional knowledge about the sets of objects and attributes. Some selection techniques use this knowledge to guide the reduction process, focusing on objects or attributes that satisfy particular constraints. For instance, some methods leverage attribute weighting [69], hierarchical structures [70], to further refine which concepts are retained. Recent efforts, such as the tri-granularity model introduced in [71], highlight a layered approach that organizes the lattice at multiple granularity levels global, local, and elementary, to systematically perform attribute reduction. Other selection methods consider the relationships between specific attributes [72], or rely on frequency thresholds and structural constraints [46], [73], [74] to highlight only the most significant concepts. Although these strategies improve upon simplistic pruning mechanisms, they often treat the selection criteria as static filters and do not fully consider the dynamic aspect of concept derivation within the lattice.

Conceptual clustering has been identified as a viable approach for concept lattice reduction. By grouping similar concepts, one can approximate or replace large sets of related concepts with fewer, representative "cluster centers," thereby simplifying the overall lattice. Traditionally, data clustering has focused on numerical datasets, leveraging geometric distance measures such as Euclidean or Manhattan distances to partition objects into meaningful clusters. However, the straightforward geometric notions of distance do not translate well to datasets characterized by categorical attributes, such as gender, location, or product categories, nor do they inherently capture the hierarchical and relational nuances of FCA-generated concepts. This limitation has prompted a surge of interest in adapting clustering methods to handle categorical data effectively [75], [76], [77]. For categorical datasets, similarity typically relies on equality checks rather than continuous-valued metrics. A simple matching measure, counting how many attributes match exactly, forms a baseline approach [78]. Yet, equality-based similarity treats all mismatches equally, ignoring subtle categorical variations and overlooking the hierarchical relationships that FCA captures [79]. Standard clustering algorithms designed for continuous vector spaces must therefore be reimagined to both accommodate categorical data and align with FCA's conceptual structures. The widely known k-means algorithm [80] exemplifies these challenges. While k-means is celebrated for its simplicity and efficiency, it cannot directly process categorical attributes without transformations that risk information loss. This shortcoming has led to the development of several k-means variants designed for categorical data. The k-modes algorithm [77], for example, replaces mean-based cluster centers with modes and employs a simple matching measure. Although k-modes can handle categorical attributes, it often faces stability issues in defining unique cluster modes and may not exploit any underlying hierarchy present in the data.

Subsequent adaptations, such as k-representative [81] and k-centers [82], introduce more refined definitions of cluster "centers." K-representative constructs representatives by considering the distribution of categorical values within a cluster, while k-centers estimates cluster centers as sets of probability distributions derived from kernel density estimations. These methods try to preserve important categorical relationships, but they still rely on vector-like representations or frequency counts that can obscure latent hierarchies or lead to

DOI: 10.14750/ME.2026.038

loss of crucial relational information. Other refined variants include fuzzy k-modes [83], scalable k-modes [84], and probabilistic k-modes [85]. These extensions enhance clustering flexibility and scalability, incorporate uncertainty modeling, and improve computation times. Yet, despite these advancements, the primary focus often remains on adapting k-means to categorical domains rather than integrating hierarchical structures like those found in FCA. Concept lattices derived from FCA inherently encode hierarchical and relational aspects that these clustering methods do not fully utilize. This gap suggests an opportunity: rather than simply clustering categorical data, we can employ the hierarchical, relational structures of FCA to guide clustering-based reduction. Formal Concept Analysis can represent data as a concept lattice, where each concept is formed by a set of attributes (intent) and a set of objects (extent). If we treat the concept lattice itself, or the underlying datasets it emerges from, as input to a clustering procedure adapted for categorical data, the resulting "cluster centers" can serve as approximations of the original concept sets. This approach can reduce the number of concepts that need to be explicitly represented, thus simplifying the lattice without completely discarding the essential information.

While these reduction techniques have advanced FCA, a notable gap remains in incorporating human language optimization principles, such as the principle of least effort [86] and Zipf's law, into lattice reduction. These linguistic insights reveal how humans naturally favor concise, high-frequency elements to minimize cognitive load [87]. By drawing on this perspective, it becomes possible to enhance both computational efficiency and cognitive accessibility in concept lattice reduction, aligning the resulting structures more closely with natural human information processing. Authors in [87], presents a pivotal study demonstrating that average information content is a superior predictor of word length in human languages compared to mere word frequency. This challenges the traditional Zipf's law, which posits that word length is primarily determined by frequency of use, with more frequent words being shorter. The authors argue that human languages optimize words lengths to achieve efficient communication by accounting for the statistical dependencies between words, aligning with principles from information theory. They introduce a formal measure of a word's average information content $I(w)$, calculated as:

$$I(w) = -\sum_{ct} P(ct|w) \log P(w|ct).$$

Where, $P(ct|w)$ is the probability of context c given word w, and $P(w|ct)$ is the probability of word w given context c. This formula captures the expected amount of information a word conveys across different contexts, reflecting its unpredictability and communicative value. Their empirical analysis across multiple languages using N-gram models reveals that words with higher average information content tend to be longer. This suggests that languages allocate longer word forms to convey more complex or less predictable meanings, thereby optimizing the balance between communicative efficiency and cognitive effort.

The authors in [88], explores the emergence of Zipf's law in human language through the lens of the principle of least effort. They propose that language evolution is driven by a trade-off between the efforts of the speaker and the hearer, leading to an optimized communication system. In their model, they represent language as a binary matrix $A = \{a_{ij}\}$, where $a_{ij} = 1$ if signal s_i refers to object r_j, and $a_{ij} = 0$ otherwise. This matrix captures the associations between a set of signals $S = \{s_1, s_2, \ldots, s_n\}$ and a set of objects $R = \{r_1, r_1, \ldots, r_m\}$. They define two key entropy measures to represent the efforts of the speaker and the hearer:

DOI: 10.14750/ME.2026.038

– Speaker's Effort: Measured by the entropy of the signal distribution, reflecting the effort in producing and retrieving signals.

$$H_n(S) = -\sum_{i=1}^{n} P(s_i) \log_n P(s_i),$$

Where $P(s_i)$ is the probability of signal s_i.

– Hearer's Effort: Measured by the average conditional entropy of objects given a signal, capturing the ambiguity from the hearer's perspective.

$$H_m(R|S) = \sum_{i=1}^{n} P(s_i)\, \mathrm{H}_m(R|s_i),$$

with,

$$\mathrm{H}_m(R|s_i) = -\sum_{i=1}^{m} P\left(r_j|s_i\right) \log_m P\left(r_j|s_i\right).$$

The authors introduce a **cost function** that combines these two efforts:

$$\Phi(\lambda) = \lambda H_m(R \mid S) + (1-\lambda) H_n(S),$$

where $\lambda \in [0,1]$ is a parameter that balances the importance of the hearer's effort versus the speaker's effort. By minimizing this cost function, they find that at a critical value λ^*, the system undergoes a phase transition. At this point, the frequency distribution of signals follows Zipf's law, indicating that efficient communication arises naturally from optimizing the balance between speaker and hearer efforts.

3.2. Summary

This chapter provided an overview of various strategies designed to manage and reduce the complexity of concept lattices in FCA. Early methods focused on removing redundant information or simplifying the lattice through algebraic or approximation techniques. While these approaches improved scalability or interpretability to some extent, they often did not fully leverage the hierarchical relationships inherent in the data or consider human cognitive factors.

As research progressed, attention turned to clustering-based approaches that more effectively handle categorical attributes and incorporate the relational structures that FCA encodes. Recent work integrates frequency, derivation costs, and insights from cognitive and linguistic studies, refining concept selection into a dynamic, cognitively aligned process. By doing so, these newer techniques achieve more human-intelligible lattice reductions, better balancing complexity, interpretability, and structural fidelity than earlier methods.

The existing body of work highlights several strategies for lattice reduction in FCA, ranging from iceberg lattices and attribute selection to clustering-based approaches. While these studies provide important insights, they also leave unresolved challenges in terms of scalability, interpretability, and cost-aware reduction. A more detailed articulation of these research gaps along with the motivations that drive the present dissertation and the core strategies it introduces is presented in the next chapter.

DOI: 10.14750/ME.2026.038

Chapter 4: Foundational Pillars of Our Proposed Strategies

4.1. Research Gaps and Foundational Strategies

As previously surveyed, many concept lattice reduction techniques have pushed the boundaries of FCA's applicability, especially in terms of computational feasibility and visual interpretability. However, critical limitations remain. Most notably, existing methods often lack a dynamic understanding of concept interrelations within the lattice structure. They may overlook derivation ease that is, how readily one concept can be derived from another a factor crucial to both algorithmic efficiency and semantic clarity. Furthermore, clustering-based FCA reductions frequently rely on geometric distance metrics ill-suited to FCA's inherently relational hierarchy, which can result in oversimplified or distorted conceptual structures.

To address these challenges, this dissertation proposes a set of novel strategies that combine foundational FCA principles with concepts from graph theory, optimization, and linguistics. These strategies are grounded in two core pillars: (1) the kernel concepts framework, and (2) the adaptation of Dijkstra's algorithm for deriving distances between concepts in the lattice. Together, these components aim to preserve the structural integrity of FCA lattices while enhancing their interpretability and scalability. While the techniques surveyed in the previous chapter have significantly advanced the quest for more manageable concept lattices, several notable shortcomings remain. Chiefly, existing methods often lack a dynamic understanding of how concepts relate within the hierarchical framework of FCA; they may also neglect how easily one concept can be derived from another. Additionally, approaches that do tackle categorical data or incorporate clustering frequently rely on distance metrics ill-suited for FCA's relational structure, leading to potential information loss.

To address these gaps, this dissertation introduces many strategies, including two novel extensions of the k-means algorithm, K-means Dijkstra on Lattice (KDL) and K-means Vector on Lattice (KVL), that aim to preserve the categorical richness and hierarchical relationships of FCA-based structures. Unlike previous methods that treat categorical attributes as flat symbols, our methods integrate the relational structure derived from FCA, using it as a guide for identifying meaningful cluster representatives, i.e., reduced concept sets. KDL exploits the lattice structure constructed from FCA by considering formal concepts as nodes and their hierarchical order as edges. Instead of relying on geometric distances, KDL uses a shortest-path computation (via a customized Dijkstra's algorithm) on the lattice to measure distances between concepts. By substituting Euclidean distance with path costs on the concept lattice, KDL identifies cluster "centroids" that faithfully represent underlying conceptual relationships, effectively capturing and preserving the data's hierarchical complexity. These centroids act as representative concepts that can replace large sets of similar concepts, thus reducing the scale of the lattice. While KDL leverages a graph-based perspective, KVL transforms each formal concept into a "concept description vector." This vectorization step, guided by FCA insights, ensures that the attributes and relationships critical to the lattice's structure are not lost. KVL then applies conventional k-means clustering to these vectors. The carefully constructed vector space retains key structural features of the lattice, enabling k-means to group related concepts into clusters. The resulting

DOI: 10.14750/ME.2026.038

cluster centers approximate the original concept sets, contributing to lattice reduction by replacing numerous related concepts with a fewer number of centroids.

Both KDL and KVL serve as reduction tools, going beyond standard clustering adaptations. They incorporate FCA's relational context to generate more interpretable, stable, and structure-preserving cluster centers. Whereas earlier categorical clustering methods focused primarily on defining suitable similarity measures or handling uncertainty, KDL and KVL integrate FCA's conceptual hierarchy to maintain the interpretability and essential properties of the concept lattice. By doing so, they transform the clustering process into a powerful lattice reduction technique.

The Kernel Concept Set (KCS) approach in this study, arises from the need to address the limitations found in traditional selection-based methods. Unlike frequency-only or attribute-centric selection techniques, the KCS method integrates both the frequency of concepts and their derivation cost. The concept of "derivation cost" introduces a crucial dynamic element: rather than merely counting how often a concept appears or which attributes it possesses, KCS evaluates how easily one concept can be derived from another, acknowledging the hierarchical and directional relationships in the lattice. This perspective enables a more holistic assessment of concept importance and interconnectivity.

One key innovation of KCS lies in its flexible derivation cost function. By refining the notion of similarity into a more general, flexible distance measure, one that can account for both usage-level patterns and internal structural details, the KCS approach transcends the limitations of static, frequency-based methods. This broader scope of application offers a nuanced understanding of concept relationships, capturing complexities that would otherwise remain hidden. Whereas earlier selection-based methods might only consider whether a concept is "frequent enough" or "fits certain attribute criteria," KCS factors in how "expensive" it is to navigate from one concept to another within the lattice, using a shortest-path interpretation influenced by Dijkstra-based measures [46].

Another distinguishing feature of the KCS approach is its capacity to identify kernel concepts that serve as conceptual "centers" or anchors. Much like centroids in clustering algorithms, these kernel concepts become reference points around which other concepts can be grouped. This reframing of concept selection as a clustering-like process separates KCS from standard selection strategies and sets it apart from conventional clustering methods that require vector spaces or rely on ad-hoc distance metrics. Instead, KCS operates efficiently within a general metric space, providing a more natural fit for the hierarchical structures inherent in FCA lattices. By doing so, it avoids the information loss commonly associated with vectorization and also foregoes the computationally expensive steps seen in standard agglomerative clustering methods.

The KCS approach, therefore, surpasses traditional methods in several critical aspects:

- It does not require a vector space; a general metric space is sufficient.
- It has a lower cost compared to the standard agglutinative clustering methods.
- Flexible distance interpretation
- It provides the cluster centroids not only the cluster members.

By combining frequency measures, derivation costs, and a robust notion of concept similarity, the KCS approach offers a more sophisticated and holistic strategy than earlier selection or simplification methods. It not only selects a minimal and representative kernel subset but also ensures that the chosen concepts form a stable backbone from which the entire lattice can be understood or reconstructed. This capability positions KCS as a novel

DOI: 10.14750/ME.2026.038

clustering methodology tailored explicitly for concept lattices, representing a significant step forward in the quest to reduce lattice complexity while maintaining meaningful and interpretable structures.

While these reduction techniques have advanced the field of FCA, there remains a gap in leveraging principles from human language optimization to enhance both computational efficiency and cognitive accessibility. Another proposed model in this study addresses this gap by drawing inspiration from first, linguistic theories that examine how human languages evolve to balance expressiveness with efficiency, such as the principle of least effort [86], a concept suggesting that humans naturally seek to minimize the amount of work they do, including in language use. Zipf observed that the frequency of word usage in a language is inversely proportional to its rank in a frequency table, a phenomenon now known as Zipf's law. This means that a few words are used extremely frequently, while the vast majority are used rarely, and second, the information theory in linguistics [87]. In human language, words and structures are optimized to convey maximum meaning with minimal cognitive load and resource expenditure.

4.2. Kernel Concepts in Concept Lattices

4.2.1. Definition of Kernel Concepts

A kernel concept in FCA is a strategically chosen formal concept within a concept lattice that serves as a pivotal "building block" for efficiently representing and deriving other concepts. The notion of a kernel concept is introduced in this dissertation as a novel reduction strategy within Formal Concept Analysis. While the idea draws inspiration from clustering principles, particularly the use of centroids in K-means, the kernel concept framework uniquely adapts this principle to the structure of concept lattices. In contrast to frequency-only reductions such as iceberg lattices [43], kernel concepts combine structural centrality, frequency, and derivation cost into a unified optimization model. To the best of our knowledge, no prior FCA work has formalized or applied this combination, making kernel concepts an original contribution of the present research. Kernel concepts are deliberately selected based on additional criteria to minimize overall complexity Typically, these criteria involve:

- Frequency: How often or how prominently a concept appears in the domain, indicating its global importance or prevalence.
- Derivation Cost**:** The computational or structural effort required to derive one concept from another, reflecting each concept's "navigational" significance in the lattice.

Formally, if C is the set of all formal concepts in a lattice and $C_M \subset C$ is a subset limited by size or cost constraints, then each concept in C_M is called a kernel concept. Together, these kernel concepts act as anchor points or centroids that can approximate or generate all other concepts with minimal overall cost. Typically, kernel concepts are subject to a *capacity constraint*, $|C_M| = S_c$, which ensures that the subset of chosen concepts does not grow too large. This limit S_c can be specified according to resource constraints, interpretability requirements, or domain-specific guidelines. The process of selecting C_M then boils down to minimizing an *aggregate cost function* such as:

DOI: 10.14750/ME.2026.038

$$min\{\sum_{c \in C} f(c)\, d(C_M, c) |\, C_M \subset C\, , |C_M| = S_c\},$$

where:

$f(c)$ indicates how "valuable" or "frequent" a concept c is,

$d(C_M, c)$ is the minimal cost to derive concept c from any concept in the kernel set C_M.

In essence, each kernel concept helps minimize the total "distance" needed to generate or approximate all other concepts while still respecting the size or capacity limit. Positioning kernel concepts among reduction methods, numerous methods exist to simplify or reduce concept lattices ranging from redundancy removal *to* abstracting hierarchies or filtering by frequency thresholds. The well-known iceberg lattice strategy, for instance, filters the concept lattice based on a single support threshold *minsupp*, typically *minsupp* $\in [0,1]$.

Formally, consider a formal context (G, M, I), where G is the set of objects, M is the set of attributes, and $I \subseteq G \times M$ is the incidence relation indicating which objects possess which attributes. For any concept (A, B) in this context, its support is measured as $supp(B) = \frac{|B'|}{|G|}$, where B' is the set of all objects in G that share exactly the attributes in B. If the support $supp(B)$ meets or exceeds the minimum threshold minsuppupp, then (A, B) remains in the iceberg lattice; otherwise, it is pruned. This straightforward criterion provides a practical means of reducing the size of the concept lattice, retaining only those concepts whose intent is sufficiently frequent in the dataset. This approach:

- Focuses purely on frequency: Only concepts that appear "often enough" are kept.
- Captures top-level groupings: The "topmost portion" of the concept lattice becomes explicit, offering a higher-level but frequency-centric view of the data.
- Does not account for derivation cost: Two concepts with the same high support remain equally in the lattice even if one concept is much more "central" for deriving other concepts.

In iceberg**,** the primary criterion revolves around the condition $supp(B) \geq minsupp$, whereby each concept's support value must exceed a fixed threshold. Consequently, the method prunes all concepts falling below that global frequency requirement. Conceptually, this can be seen as filtering the lattice down to its "topmost" or "most frequent" portion. However, this design choice also entails a caveat: while it efficiently isolates those concepts that appear very often in the data, it may discard concepts with lower support that are structurally pivotal in deriving or relating other parts of the lattice. Moreover, no inherent metric accounts for derivation cost or traversal complexity.

In contrast, kernel selection depends on a more nuanced blend of concept frequency $f()$ and derivation $cost\ d()$. By focusing on a combined or weighted measure of usage and "hubness," kernel selection aims to locate a small but influential set of "centroid" concepts from which the entire lattice can be efficiently derived or approximated. This balanced approach provides a richer interpretation, ensuring each chosen concept is both sufficiently frequent and well-connected within the conceptual structure. Naturally, one trade-off is that it demands an auxiliary cost metric $d()$, adding a layer of optimization that typically involves more complex computations than a simple frequency threshold.

DOI: 10.14750/ME.2026.038

4.2.2. Role and Importance of Kernel Concepts

Kernel concepts serve as the linchpin in effectively managing, interpreting, and streamlining large concept lattices within FCA. Their selection, driven by criteria such as frequency and derivation cost, brings multiple advantages:

1. Structural Backbone: By design, each kernel concept often functions as a "hub" for deriving or approximating numerous other concepts. This positions kernel concepts as the structural backbone of the lattice, ensuring that the essential relationships and crucial data patterns remain intact even after significant reductions in overall lattice size.
2. Computational Efficiency: Concept lattices can grow exponentially with the size of the dataset, imposing high computational and memory demands. Identifying a minimal kernel set that still covers or approximates the entire lattice substantially reduces computational overhead. In many cases, one can generate or retrieve non-kernel concepts on-demand from kernel concepts through derivation, thus avoiding explicit enumeration of all possible concepts.
3. Balanced Criterion Beyond Frequency Alone**:** Simple thresholds (e.g., iceberg approaches) focus on frequency, risking the exclusion of structurally pivotal but less frequent concepts. Kernel concepts, however, account not just for how often a concept appears (its frequency) but also for how readily (or "inexpensively") it can serve as a representative. This dual perspective often yields a more faithful representation of the lattice's inherent relationships, balancing global relevance with local connectivity.
4. Interpretability and Usability: Large and dense concept lattices can overwhelm users seeking patterns or insights. By highlighting the kernel subset, analysts can more easily navigate "anchor points" within the data, making subsequent visualization, exploration, or domain-specific interpretations more straightforward. This enhanced clarity is crucial in fields such as knowledge discovery, ontology learning, and database marketing, where decision-makers must interpret and act on complex data structures.
5. Facilitating Further Analysis: Kernel concepts frequently serve as natural "centroids" or "cluster centers" in conceptual clustering and approximation tasks. Once determined, they can be integrated into downstream workflows such as generating bases of association rules or supporting user queries without recomputing or storing the full lattice. This modularity fosters efficient iterative analyses, allowing repeated refinement or extended exploration of the data's conceptual organization.

In essence, kernel concepts play an essential role in bridging the gap between complete conceptual representation and practical scalability. They constitute a carefully chosen subset that simultaneously conserves the key structural and semantic properties of the lattice and promotes more efficient knowledge processing.

4.3. Dijkstra's Algorithm in Concept Lattice Reduction

4.3.1. Background and Motivation

Dijkstra's algorithm, introduced in 1959 by Edsger W. Dijkstra, is a cornerstone method for computing shortest paths in directed, weighted graphs. Its efficiency and general

DOI: 10.14750/ME.2026.038

applicability have led to its adoption across multiple fields, including Internet routing where it determines optimal data traversal paths between network nodes, and various transportation and logistics applications that require identifying fast, cost-effective routes [89]. Given a directed weighted graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, each edge e∈E has a non-negative weight representing its traversal cost. Dijkstra's algorithm systematically calculates the shortest path from a source vertex to every other vertex in V, offering a reliable solution to a wide range of shortest-path problems.

The algorithm marks vertices as either "temporary" or "visited," continually updating tentative distances from the source. It terminates once all vertices have been processed. However, the algorithm cannot handle negative edge weights directly, potentially limiting its accuracy if such edges are present [90]. Another practical consideration is the choice of data structures for managing priority queues, which influences the algorithm's time complexity:

- Using a Fibonacci Heap: Complexity:

$$O(|V|\log|V| + |E|).$$

  In this case, DeleteMin operations take $O(1)$ amortized time, providing theoretically optimal performance.

- Using a Standard Binary Heap: Complexity:

$$O(|E|log|V|).$$

  Here, the algorithm performs |E| updates for the standard heap, typically yielding efficient performance in many real-world scenarios.

- Using a Priority Queue (e.g., array-based): Complexity:

$$O(|V|^2).$$

  This arises from repeated scans of the unordered set New Frontier, up to $|V|$ times to find the vertex with the minimum temporary distance (sDist) value.

Beyond the standard Dijkstra's method, several variants cater to specific conditions. The Bellman-Ford algorithm [91] accommodates negative-weight edges at a higher computational cost. The Floyd-Warshall algorithm [92] uses dynamic programming to manage both positive and negative weights comprehensively. Johnson's algorithm [93] employs Bellman-Ford to reweight edges, eliminating negatives and reducing execution time for sparse graphs. The A* algorithm integrates heuristics with breadth-first search principles, potentially increasing efficiency in certain contexts, albeit with some risk to completeness or absolute accuracy [94]. The appropriate choice among these methods depends on factors such as graph density, edge weight properties, and performance requirements.

However, when applying FCA to complex categorical datasets, traditional Euclidean or frequency-based distance measures often fail to reflect the nuanced hierarchical relationships encoded in the concept lattice. To address this gap, adapting Dijkstra's algorithm to measure distances within the lattice proves advantageous. In this adapted view:

DOI: 10.14750/ME.2026.038

1. Vertices (Nodes) become formal concepts derived from the FCA context.
2. Edges represent hierarchical relationships (e.g., the partial order $\leq$ between concepts), with assigned weights corresponding to upward or downward moves in the lattice.

This integration ensures that path costs capture not just the frequency of concepts (as in simpler pruning methods) but also their relative "distance" or "effort" within the lattice's structure.

4.3.2. Dijkstra-Based Distance in FCA

The Dijkstra-based distance measure plays a pivotal role in our proposed frameworks. Providing a more suitable alternative to the conventional metrics, Dijkstra's algorithm operates directly on the concept lattice derived from categorical data through FCA. By incorporating the inherent structure of the lattice, the method considers direction-sensitive costs, typically assigning a higher cost to upward (parent-to-child) movements than to downward (child-to-parent) transitions. This directional weighting more accurately reflects the hierarchical nature of the data. To enhance efficiency, the algorithm employs a min-heap-based priority queue, ensuring that calculations for shortest paths, are performed both effectively and with minimal computational overhead.

Formally, consider a concept lattice $\mathcal{B}(C, <)$, and its corresponding graph $\mathcal{H}(C, E)$, where C represents the set of formal concepts and E denotes the edges signifying hierarchical relationships. Let C_s and C_e be two distinct formal concepts in C, with C_s serving as the starting point and C_e as the endpoint for the path calculation. Each concept $c \in C$ has an associated cost $d(c)$ that represents the cost of reaching c from C_s. To differentiate the directionality of traversal along the lattice edges, two cost parameters are defined: "UpCost" for moving from a concept to a more specific (child) concept, and "DownCost" for moving from a concept to a more general (parent) concept.

Within this framework, the Dijkstra-based distance measure relies on a priority queue Q, implemented as a min-heap keyed by $d(c)$, and a set V tracking visited nodes. The cost function $f: C \times C \rightarrow \mathbb{R} \cup \{\infty\}$ evaluates the cost of moving from one concept c to an adjacent concept c′ based on their relation:

$$f(c, c') = \begin{cases} UpCost, & if\ c \supseteq c', \\ DownCost, & otherwise. \end{cases}$$

Combining these costs over a sequence of concepts forms the basis for calculating the shortest path. Thus, for all paths (c_1, c_2, …, c_n) from C_s to C_e, the Dijkstra-based distance measure $d(C_s, C_e)$ selects the path with the minimal cumulative cost:

$$d(C_s, C_e) = min\left\{ \sum_{i=1}^{n-1} f(c_i, c_{i+1}) \mid (c_1, c_2, \dots, c_n) \text{ is a path from } C_s \text{ to } C_e \right\},$$

Here, the measure $d(C_s, C_e)$ represents the minimal cost required to navigate the lattice from the starting concept C_s to the target concept C_e, effectively encapsulating both the structure of the concept lattice and the directional constraints inherent in the data's hierarchy. The algorithm functions as follows:

DOI: 10.14750/ME.2026.038

Algorithm 4.1: The Dijkstra-Based Distance Measure Algorithm on The Concept Lattice

Inputs: C_s, C_e, $\mathcal{H}(C, E)$, UpCost, DownCost.
Output: minimum cost from C_s to C_e
Initialize:
    $For\ each\ c\ in\ \mathcal{H}$:
        $d(c) \leftarrow \infty$
    $EndFor$
    $d(C_s) \leftarrow 0$
    $Initialize\ P\ and\ V \leftarrow \emptyset$
    $Insert\ (0, C_s)$ into Q
While $Q \neq \emptyset\ do$:
    $(d(c), c) \leftarrow Dequeue(Q)$
    $If\ c = Ce\ then$:
        Return $d(C_e)$.
    $EndIf$
    $If\ c\ not\ in\ V\ then$:
        $Add\ c\ to\ V$
    $EndIf$
    $For\ each\ neighbor\ u\ of\ c\ do$:
        $If\ \ neighbor\ not\ in\ V$:
            $If\ c\ is\ a\ superset\ of\ u\ then$:
                $cost \leftarrow d(c) + UpCost$
            $Else$:
                $cost \leftarrow d(c) + DownCost$
            $EndIf$
            $If\ cost < d(u)\ then$:
                $d(u) \leftarrow cost$
                $P(u) \leftarrow c$
                $Enqueue\ (d(u), u)\ into\ Q$
            $EndIf$
        $EndIf$
    $EndFor$
$EndWhile$

In the presented framework, C_s represents the starting concept from which the shortest path calculation begins, and C_e designates the target concept. The structure $\mathcal{H}(C, E)$ symbolizes the concept lattice, comprising the set of concepts C and their connecting edges E. Within this framework, UpCost and DownCost are predefined metrics quantifying the cost of transitioning upward or downward along the lattice edges. The shortest path distances from C_s to any given concept c are stored in $d(c)$, while a predecessor map P indicates the immediate predecessor of c along the shortest path, ensuring a traceable route from C_s to C_e. The priority queue Q manages pending concepts to explore, and the set V records already visited nodes. Due to the lattice's inherent connectivity, the algorithm always identifies a path between C_s and C_e.

The time complexity of this approach, $O(E + C\ log(C))$, reflects the interplay of the number of edges E and concepts C, combined with efficient operations on the min-heap-based priority queue. By capitalizing on the lattice's structured relationships and integrating directionally-sensitive cost functions, this Dijkstra-based distance measure more precisely captures categorical dissimilarities. Consequently, it supports a more streamlined clustering procedure and improves the accuracy and interpretability of the resulting cluster assignments.

By adapting Dijkstra's algorithm to work directly on concept lattices, we achieve a *structure-aware* distance measure that elevates categorical data analysis beyond simpler frequency-based or geometric approaches. Each path cost reflects not just how often a concept appears or how large its extent might be, but also how *structurally central* it is. This distance measure undergirds various parts of our proposed FCA-based reduction framework:

DOI: 10.14750/ME.2026.038

it guides kernel concept selection, steers the clustering of concepts, and helps maintain a manageable yet conceptually rich representation of high-dimensional, complex datasets.

4.4. Baseline Greedy Algorithm for Kernel Concepts Selection

This section introduces a *baseline* Greedy Algorithm for identifying a kernel concept set within a large formal concept lattice. While simpler and less optimized than the advanced methods detailed in subsequent chapters, this algorithm demonstrates how integrating concept frequency and derivation costs can produce a smaller, yet structurally significant, subset of concepts. It selects the most beneficial concept at each step based on two key measures: frequency and derivation cost. Despite its simplicity, the algorithm can become time-consuming when applied to larger datasets or kernel sizes, highlighting the necessity for more advanced or optimized approaches.

4.4.1. Kernel Concepts Selection

In the kernel concept selection process, we focus on two measures for each concept $c \in C$ (the set of all concepts in the lattice):

1. Frequency $f(c)$
   A positive real-valued function

   $$f : C \rightarrow \mathbb{R}^{+},$$

   quantifying how relevant or frequently a concept c appears. This metric highlights concepts that are crucial within the domain.

2. Derivation Cost $d()$
   A function

   $$d : C \times C \rightarrow \mathbb{R}^{+} \cup \{0\},$$

   indicating the "cost" of deriving one concept from another within the lattice.

   - Self-Cost: $d\ (c, c) = 0$, for any concept c within the lattice, indicating no cost for self-derivation.
   - Asymmetric Cost: For two different concepts c_1 and c_2, $d(c_1, c_2) \neq d\ (c_2, c_1)$, reflecting the directional nature of derivation within the lattice.
   - Integration of Dijkstra-Based Distance Measure: To refine the calculation of asymmetric costs between concepts, we have employed the Dijkstra-Based Distance Measure from. This approach computes the shortest path in the lattice considering the direction and cost of the path. Specifically, we have set the cost for upward transitions (parent-to-child) in the lattice as 2 and for downward transitions (child-to-parent) as 1. This integration adds a layer of sophistication to our function d, allowing it to more accurately represent the complexities involved in navigating the concept lattice.

3. Frequency-Weighted Derivation Cost

For a subset of concepts $K_c \subseteq C$, let

DOI: 10.14750/ME.2026.038

$$d\ (K_c, c)\ =\ min\ _{c_i \in K_c}\{d\ (c_i, c)\ |\ c_i \in K_c\}.$$

then the frequency-weighted derivation cost becomes:

$$d^{\,f}(K_c, c) =\ f(c) \cdot\ d(K_c, c).$$

This expression captures both (a) how important c is, and (b) how far c lies from the chosen set C_s.

4. Kernel Concept Set: Optimization Constraint

We seek a kernel set $K_{\min}$ of maximum size S_c that minimizes the sum of frequency-weighted distances over *all* concepts:

$$K_{min}=argmin_{K_c \subset C}\{\sum_{c\ \in C} d^{\,f}(K_c, c)|\ |K_c| =\ S_c\}.$$

In simpler notation, define:

$$AggCost(K_c) = \sum_{c \in C} (f(c) \cdot\ d(K_c, c)).$$

We want K_{min} such that $AggCost(K_c)$ is minimized and $|K_c| =\ S_c$.

4.4.2. Baseline Greedy Algorithm Steps

A greedy approach offers a direct, though not always optimal, way to find a suitable kernel set. Below are the main steps:

Algorithm 4.2: Baseline Greedy Concepts Kernel Selection

Input:

- The set of all concepts C
- Frequency Value for each $c\ \in C$
- Maximum Core Set Size S_c (maximum kernel set size)
- Transition Cost: upward ← 2, downward ← 1

Output:

- Kernel Concept Set K_C

Algorithm Steps:

1. Initialization:
   - Initialize $K_C \leftarrow None$.
2. Derivation Cost Calculation:
   - For each concept c in the lattice, calculate the minimal derivation cost to every other concept using Dijkstra's algorithm. Apply the Dijkstra-Based Distance Measure, for asymmetric cost calculation between concepts, as:

$$d\ (K_c, c)\ =\ min\ _{c_i \in K_c}\{d\ (c_i, c)\ |\ c_i \in K_c\}$$

3. Aggregated Derivation Cost Computation:

For a given subset $K_c \subset C$, calculate the aggregated derivation cost using the formula:

$$AggCost(K_c) = \sum_{c \in C} (f(c) \cdot\ d(K_c, c))$$

4. Kernel Set Identification:
   - Define S_c, the maximum size for the Kernel set.

DOI: 10.14750/ME.2026.038

- Initialize best_cost ← (∞), best_candidate ← None.
- Iteratively add concepts to K_C using a greedy algorithm approach:
  - Select the concept that most reduces the aggregated derivation cost.
  - Update best_cost and best_candidate as optimal options are found.
  - Continue until $|K_C|=|S_c|$ or no further reduction in the aggregated derivation cost is possible.

5. Result Analysis:
   - The final K_C represents the kernel concept set that minimizes the aggregated derivation cost, adhering to the constraint $|K_C|=|S_c|$.

One major bottleneck in the baseline Greedy Algorithm is the repeated shortest-path derivation cost calculation, typically carried out via Dijkstra's algorithm at $O(V^2)$ per concept (where V denotes the total number of concepts). If performed naively for every pair of concepts, this cost may inflate to $O(V^3)$. Consequently, the baseline method can become prohibitively slow on large or dense lattices, underscoring the need for more efficient or optimized approaches in practical FCA scenarios.

4.4.3. Experimental Setup and Methodology

4.4.3.1. Impact of Kernel Set Size on Derivation Cost and Execution Time

In our study, we analyzed how the baseline greedy algorithm in FCA responds to varying kernel concept set sizes across multiple benchmark datasets. Specifically, we experimented with four widely used datasets from the UCI Machine Learning Repository Balance-Scale[1], Breast Cancer Wisconsin[2], Teaching assistant evaluation (Tae)[3], and Car Evaluation[4]. For each dataset, the kernel set size was systematically adjusted between 15% and 30% of the total number of concepts in its corresponding lattice (as summarized in Table 4.1). This

1. Derivation Cost
   - Reflects the aggregated resources needed to derive all relevant concepts once the kernel set is chosen.
   - We hypothesize that increasing the kernel concepts set size, thus encompassing more concepts in the core set, simplifies the structure and lowers derivation cost**.**

2. Runtime
   - The time required by the greedy algorithm to identify the kernel set.
   - As the kernel set size grows, we expect more steps and candidate checks, leading to higher runtime.

To ensure robustness, each lattice configuration in Table 4.1 is tested multiple times, varying the kernel set proportion (15%, 20%, 25%, 30% of total concepts). The results are captured (Figure 4.1), illustrating how changes in kernel size affect derivation cost and runtime.

Table 4.1. Lattice Characteristics

Formal Contexts	#Object	#Attributes	Density	# Formal concepts	#Edges
Balance-Scale	625	20	0.18	1070	3822

1 Balance-Scale dataset: https://archive.ics.uci.edu/dataset/12/balance+scale.

2 Breast Cancer dataset: https://archive.ics.uci.edu/dataset/14/breast+cancer.

3 Tae Dataset: https://archive.ics.uci.edu/dataset/100/teaching+assi stant+evaluation.

4 Car Evaluation dataset: https://archive.ics.uci.edu/dataset/19/car+evaluation.

DOI: 10.14750/ME.2026.038

Breast Cancer	286	43	0.20	2132	7818
Tae	151	101	0.05	276	619
Car Evaluation	1728	21	0.20	3596	14917

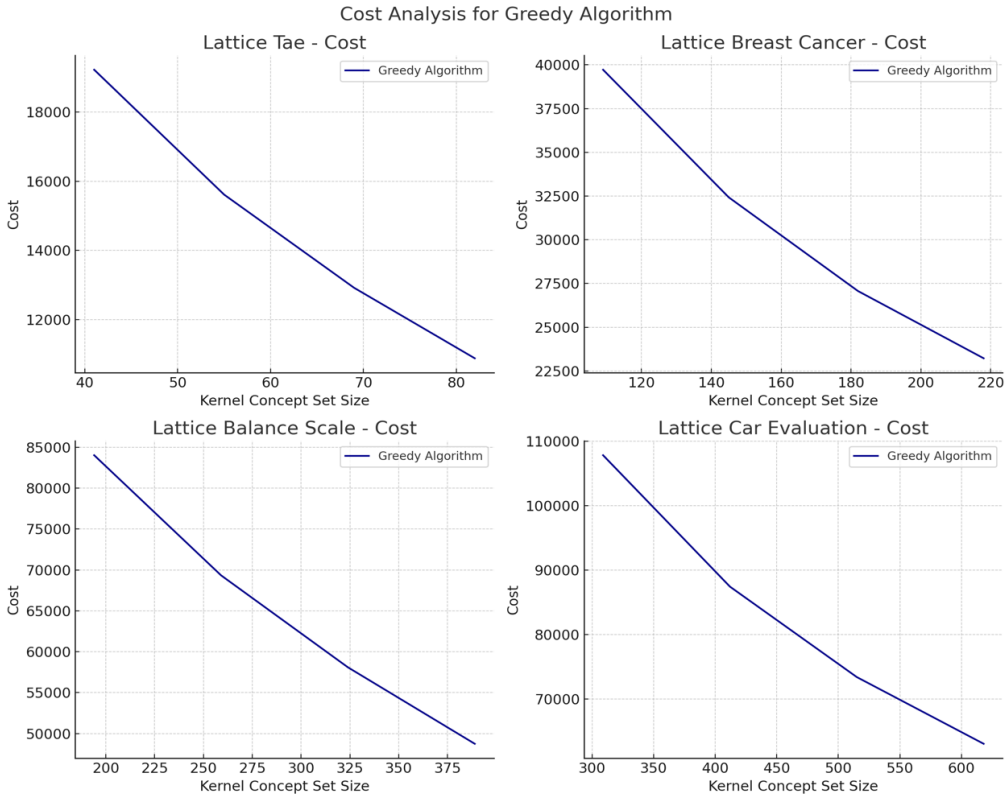


Figure 4.1. Cost Analysis for Greedy Algorithm Across Kernel Concept Set sizes

As shown in Figure 4.1, the experimental results reveal that enlarging the kernel set consistently reduces derivation cost, indicating that a more encompassing core set effectively simplifies the effort required to derive remaining concepts. However, this advantage is counterbalanced by growing computational demands: each added concept triggers more candidate checks, causing runtime to increase noticeably at higher kernel sizes as shown in Figure 4.2. Although the baseline greedy method remains a practical solution for moderate datasets, its scalability begins to wane when kernel proportions approach 30%, especially in contexts with many formal concepts (e.g., Car Evaluation).

4.4.3.2. Impact of Lattice Size on Derivation Cost and Runtime

We further examine the baseline greedy algorithm by varying the size and complexity of the lattices themselves, as detailed in Table 4.1. This step explores how the number of formal concepts and overall lattice density affect two main metrics:

1. Derivation Cost: The aggregated effort required for concept derivation once the kernel is chosen.
2. Runtime: The total time the baseline greedy method takes to select a kernel set of fixed proportion (e.g., 30%) from the lattice.

In particular, Figure 4.3 visually captures these relationships across the four different datasets from Table 4.1. When considering smaller lattices (e.g., Breast Cancer or Tae, each

DOI: 10.14750/ME.2026.038

with fewer than a thousand formal concepts), the baseline greedy algorithm strikes a reasonable balance between lowering derivation cost and keeping runtime manageable. However, with larger lattices (e.g., Car Evaluation, featuring over two thousand formal concepts), runtime escalates rapidly, highlighting the baseline algorithm's limited scalability.

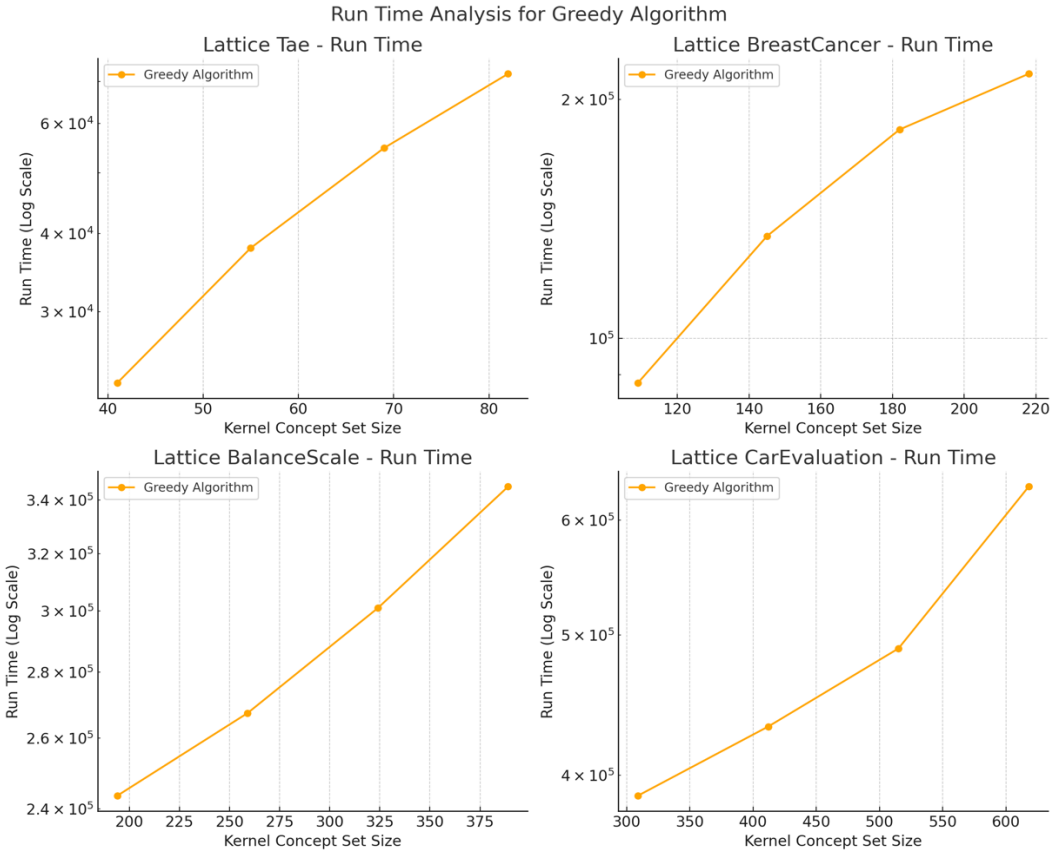


Figure 4.2. Runtime Analysis for Greedy Algorithm across Kernel Concept Set sizes

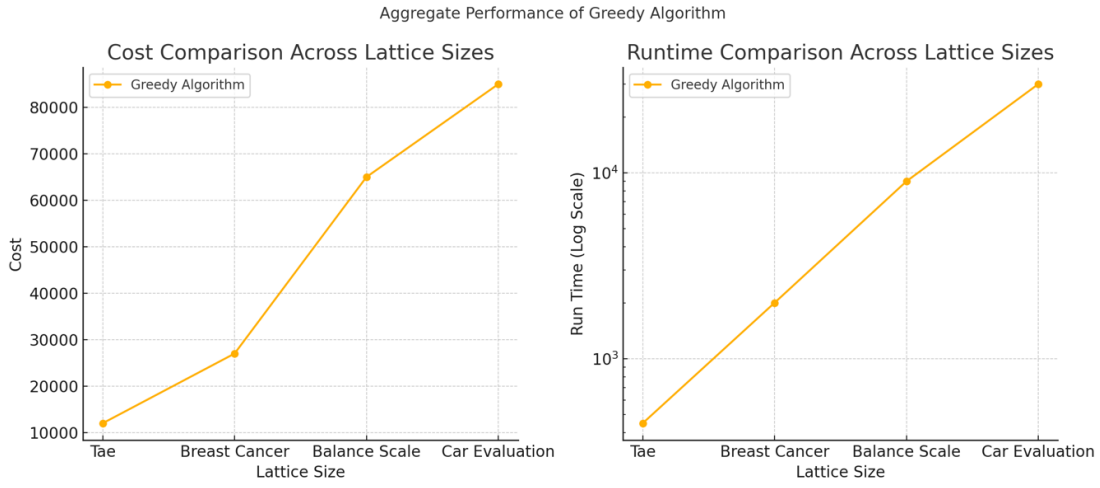


Figure 4.3. Performance Analysis of the Baseline Greedy Algorithm on Derivation Cost and Runtime Across Different Lattice Sizes

DOI: 10.14750/ME.2026.038

From these results, several observations emerge:

- Stable Performance on Small Lattices: As seen in Figure 4.3 for Breast Cancer or Tae, when the lattice has fewer concepts, the baseline greedy approach effectively reduces derivation cost with minimal runtime growth.
- Sharp Runtime Increases in Larger Lattices: For datasets like Car Evaluation, Figure 4.3 shows a more dramatic rise in runtime, suggesting that repeated cost computations and candidate checks become exponentially more expensive**.**
- Consistent Reduction in Derivation Cost: Regardless of lattice size, the method reliably lowers derivation cost particularly beneficial in moderately sized lattices though the runtime penalty intensifies in bigger ones.

Overall, Figure 4.3 underscores that while the baseline greedy algorithm adeptly decreases derivation cost across the studied datasets, it becomes noticeably slower for extensive lattices containing large numbers of formal concepts. These findings confirm that the baseline approach remains a viable choice for small to medium lattice sizes but may require optimization or alternative methods to maintain feasible runtimes in large-scale FCA applications.

4.5. Summary

This chapter presented the essential building blocks of the proposed reduction framework, namely kernel concepts, Dijkstra-based distance calculations, and a baseline Greedy Algorithm. These components demonstrated how structural properties of the lattice could be leveraged to achieve more compact and interpretable representations. The greedy approach, in particular, showed that combining frequency measures with derivation costs made it possible to select a smaller but meaningful subset of concepts that preserved the lattice's structural backbone. Even as a simple heuristic, it highlighted the usefulness of kernel concepts as pivotal anchors for both efficiency and clarity.

At the same time, the limitations of the baseline approach were clear, especially its lack of scalability and high runtime on larger lattices. These constraints underscored the need for more advanced strategies and provided the motivation for the techniques developed in the following chapters. By laying this groundwork, the chapter established both the feasibility and the challenges of lattice reduction, setting the stage for optimized and cognitively aligned methods that extend beyond the greedy framework while ensuring tractability and interpretability for large-scale FCA applications.

DOI: 10.14750/ME.2026.038

Chapter 5: Clustering-Based Reduction Strategies for FCA

5.1. Introduction

Concept management is a key dimension of knowledge engineering, where ontologies play a central role in structuring and representing domain-specific knowledge [95]. Widely adopted standards such as Resource Description Framework (RDF), the Web Ontology Language (OWL), SPARQL Protocol and RDF Query Language (SPARQL), and Description Logic [96] provide powerful tools for describing concepts, relationships, and constraints. However, these frameworks typically rely on human experts to define atomic concepts and derivation rules, offering limited automation for concept generation from raw data.

Beyond ontologies, automated concept generation is pursued through various analytical approaches. Two prominent families are Conceptual Clustering (CC) and FCA. Conceptual clustering [97] partitions unlabeled objects into meaningful clusters, each described by conceptual patterns or attributes. Traditional conceptual clustering methods often rely on numerical taxonomies and distance-based measures. While effective for numerical features, these techniques face substantial limitations when dealing with categorical data. The resulting clusters may not be well-characterized in intuitive, human-readable conceptual terms [98].

To address these shortcomings, multiple variants of conceptual clustering have emerged. These include [47]:

- Distance optimization methods: These approaches start with an initial set of clusters and incrementally refine them by minimizing a predefined distance-based objective function. At each step, elements may be reassigned to different clusters if such a move leads to a lower overall cost. The algorithm iterates this process until it reaches a stable configuration where no further improvement can be made. The resulting clusters are thus formed by continuously optimizing for minimal intra-cluster distances, often leading to well-defined groupings that reflect the underlying data structure.
- Interesting-pattern discovery methods: In these techniques, the focus shifts from purely geometric measures to identifying significant recurring patterns within the data. Methods inspired by frequent itemset mining [99] search for commonly co-occurring attribute values across different objects. By filtering out infrequent or irrelevant patterns, the algorithm highlights the most characteristic and discriminative features of clusters. Consequently, concepts and clusters emerge from these frequent patterns, providing richer semantic descriptions than those relying solely on numeric similarity.
- Tree-based approaches: Tree-based conceptual clustering techniques, such as RUMMAGE [98], employ a hierarchical partitioning strategy. The dataset is recursively split into subsets based on the values of certain attributes, effectively building a conceptual tree structure. At each branching point, an attribute or attribute-value condition forms a “conceptual description” for the subsets. This top-down approach ensures that the resulting clusters are not only distinct from one another but also described by meaningful, interpretable attribute-based rules.

DOI: 10.14750/ME.2026.038

- Evolutionary strategies: Evolutionary approaches like the Multiobjective Evolutionary Conceptual Clustering Methodology (EMO-CC) [100] apply bio-inspired techniques, such as genetic algorithms, to guide the clustering process. Candidate clusterings are represented as individuals in a population. Through operations akin to mutation and crossover, as well as selection pressures favoring clusters with desirable properties (e.g., compactness and interpretability), the method evolves increasingly refined clusterings over time. The multiobjective aspect accommodates simultaneous optimization of multiple criteria, balancing various quality measures to yield conceptually rich and well-organized clusters.
- Statistical methods: Statistical-based methods, exemplified by COBWEB [101], incrementally form a hierarchical classification tree by adding objects one at a time. Each node in the tree corresponds to a probabilistic concept—a distribution of attribute values—that reflects a particular class of objects. A heuristic measure known as category utility guides the tree growth and partitioning decisions. Category utility rewards partitions that improve the predictive power of attribute values for object classification. This probabilistic and heuristic-driven approach results in a tree of concepts that are both statistically coherent and conceptually meaningful, enabling intuitive comprehension of the data's structure.

In practice, the widely acclaimed k-means algorithm [80] excels in simplicity and efficiency, particularly for large numerical datasets. However, its direct application to categorical data is problematic. Adaptations like k-modes [77], k-representative [81], and k-centers [82] have been proposed. While these adaptations can handle categorical data by redefining "cluster centers" and "similarity measures," they often require data transformations that risk losing hierarchical relationships inherent in the data. On another front, FCA models data as objects and binary attributes, producing a concept lattice that captures all possible formal concepts. Despite offering a comprehensive view, the resultant concept lattice can be excessively large, complicating both computation and interpretability. Efficient reduction of concept lattices, preserving only essential concepts, is a key research direction. Moreover, real-world concepts seldom derive from a single consistent attribute set. Instead, multiple attribute subsets might characterize a concept under different conditions, suggesting that non-crisp, flexible construction methods would be beneficial.

Integrating the strengths of conceptual clustering and FCA presents new opportunities. Conceptual clustering can manage object partitions efficiently, while FCA provides a structured representation of hierarchical concept relationships. Merging these approaches demands methods adept at categorical and hierarchical data handling, which is where our contributions lie, and specifically, we leverage the idea of extracting a smaller set of "centroids" or *kernel* concepts to achieve effective lattice reduction while retaining essential structure.

This chapter introduces two novel extensions of the k-means algorithm for categorical data within the FCA framework, K-means Dijkstra on Lattice (KDL) and K-means Vector on Lattice (KVL), with the overarching goal of reducing the original concept lattice to a smaller kernels subset. These methods aim to integrate hierarchical conceptual structures with efficient clustering, thereby retaining critical relationships while improving manageability and interpretability:

- KDL leverages FCA to construct a graph of formal concepts and employs a customized Dijkstra algorithm to measure distances within this lattice. It identifies

DOI: 10.14750/ME.2026.038

centroids (kernels) that are formal concepts with minimal intra-cluster distance, accurately capturing hierarchical and categorical relationships.
- KVL, on the other hand, translates formal concepts into numerical concept description vectors, applying traditional k-means for scalability at the possible expense of overlooking some hierarchical nuances.

By combining these methods with a parameterized distance function, we gain flexible control over the size of the resulting concept hierarchy and the degree of approximation it provides. Notably, this approach also enables an approximate reduction of formal concept lattices, resulting in a more manageable, kernel-based view of the data. In essence, KDL and KVL bring together conceptual clustering and FCA, automating the generation of a streamlined concept set that preserves interpretability and remains grounded in human-centric conceptualization.

5.2. K-Means Algorithm and Its Extensions

The k-means algorithm [80] is a well-known partitional clustering technique frequently employed in various data analysis scenarios. It assumes a dataset D composed of N numerical objects and aims to separate them into k non-empty, disjoint clusters (with $k \leq N$). A core objective of k-means is to minimize the total within-cluster variation, typically measured as the sum of squared distances from each object to the center (centroid) of the cluster it belongs to.

Mathematically, k-means can be framed as an optimization problem. Let $U = [u_{i,j}]$ denote the partition matrix, where $u_{i,j}$ is a binary indicator that specifies whether object X_i is assigned to cluster S_j. Let $Z = \{Z_1, Z_2, \ldots, Z_k\}$ represent the set of cluster centers. Typically, the squared Euclidean distance dis(X_i, Z_j) [102] is used to quantify how far each data point is from a given cluster center.

The cost function to be minimized, $P(U, Z)$, is given by:

$$P(U,Z) = \sum_{j=1}^{k}\sum_{i=1}^{N} u_{i,j}\, dis\big(X_i, Z_j\big).$$

This minimization is subject to constraints ensuring that each object X_i belongs to exactly one cluster:

$$\sum_{j=1}^{k} u_{i,j} = 1, \qquad 1 \leq i \leq N,$$
$$u_{i,j} \in \{0,1\}, 1 \leq i \leq N, 1 \leq j \leq k.$$

Here, $u_{i,j} = 1$, if X_i is assigned to S_j, and 0, otherwise. The cluster centers Z_j correspond to the mean positions of objects assigned to that cluster.

The k-means algorithm proceeds iteratively through four main steps:
- Initialize cluster centers as $Z^0 = Z_1^0, \ldots, Z_k^0$, and set $t = 0$.
- With fixed cluster centers Z^t, solve $P(U, Z^t)$ to obtain partition matrix U^t. Each object X_i is assigned to the cluster with the nearest cluster center.

DOI: 10.14750/ME.2026.038

– With fixed partition matrix U^t, generate updated cluster centers Z^{t+1} to minimize P (U^t, Z^{t+1}). The new cluster centers are computed as the mean of the objects within each cluster.

– If convergence is reached or a stopping criterion is satisfied, output the final result and terminate. Otherwise, increment t by 1 and go back to step 2.

By repeatedly adjusting both cluster memberships and centers, k-means converges to a local minimum of the objective function. Although highly effective for numerical data, k-means encounters difficulties when dealing directly with categorical data. The fundamental challenge lies in the absence of a natural numeric representation and a meaningful way to compute means or geometric distances for categorical variables. This motivates the development of specialized extensions and adaptations of k-means to handle categorical data more appropriately.

A range of extensions and modifications have been introduced to the k-means algorithm to enable its application to categorical data. One of the most prominent approaches is the K-modes algorithm [77]. In contrast to the original k-means method, which depends on Euclidean distance metrics, K-modes utilizes a dissimilarity measure specifically designed for categorical attributes. Instead of treating data as points in a Euclidean space, K-modes employs a simple matching distance and defines "cluster centers" in terms of modes rather than means.

For two categorical objects X and Y, each described by M categorical attributes, the K-modes dissimilarity is computed by counting how many attribute values differ. Formally:

$$dis(X,Y) = \sum_{i=1}^{M} \delta(X_i, Y_i),$$

with

$$\delta(X_i, Y_i) = \begin{cases} 0, & if\ X_i = Y_i\ , \\ 1, & if\ X_i \neq Y_i. \end{cases}$$

Within a cluster composed of N categorical objects $\{X_1, ..., X_N\}$, where $X_i = (x_{i1}, ..., x_{iM})$ and $1 \leq i \leq N$ is determined by selecting the most frequently occurring category for each attribute position m ($1 \leq m \leq M$) across the cluster's objects $\{x_{1m}, ..., x_{Nm}\}$. These alterations, introduced in [77], adapt the clustering process for categorical data while preserving the fundamental iterative nature of k-means. Nonetheless, it is important to note that the mode for a given cluster may not be unique, potentially introducing variability in the clustering outcome based on how modes are chosen.

For a given cluster composed of categorical objects $\{X_1, ..., X_N\}$, with each object represented as $X_l = (x_{i1}, ..., x_{iM})$ for $1 \leq i \leq N$, the K-modes algorithm determines the cluster's mode $Z = (o_1, \ldots, o_M)$ by selecting o_m, $1 \leq m \leq M$, as the attribute value that occurs most frequently in the set $\{x_{1m}, ..., x_{Nm}\}$. This approach, introduced by the authors in [77], adapts the standard K-means framework to handle categorical attributes by replacing numerical means with modes. However, it is important to note that a given cluster's mode may not be uniquely defined, multiple attribute values can share the same highest frequency. This potential ambiguity can introduce instability into the clustering process, as the final

DOI: 10.14750/ME.2026.038

outcome may depend on the particular mode chosen from among several equally frequent candidates.

The k-Representative algorithm [81] represents a further adaptation of the K-means framework, introducing the concept of cluster representatives to handle categorical data. Instead of using a single mode to characterize a cluster, as done in K-modes, the k-Representative approach defines a representative that captures the distribution of attribute values within the cluster.

Consider a cluster S consisting of p categorical objects: $S = \{X_1, \ldots, X_p\}$., where each object (x_{i1}, ..., x_{iM}) with the condition $1 \leq i \leq p$, and each x_{iM} corresponds to the value of the m-th attribute. For each attribute m ($1 \leq m \leq M$), we define O_m^S as the set of distinct categorical values that attribute m can take within cluster S. In other words, O_m^S = $\{o_{m_1}, ..., o_{m_l}\}$, where each o_{m_1} is a unique category observed in the m-th attribute across all objects in S.

Consider a cluster S composed of p categorical objects: $S = \{X_1, \ldots, X_p\}$, where each object X_I = (x_{i1}, ..., x_{iM}) for $i = 1, \ldots, p$, and M denotes the number of attributes. For each attribute m ($1 \leq m \leq M$), we define O_m^S as the set of all distinct categorical values that the m-th attribute takes on within cluster S. In other words, O_m^S is derived by examining the m-th attribute values $\{x_{1m}, ..., x_{pm}\}$ of every object in S and collecting the unique categories observed. This set O_m^S thus represents all the different categorical values that attribute m can assume across the entire cluster S.

For example, consider a cluster S containing three objects:

- Object 1: (Red, Circle, Large)
- Object 2: (Blue, Circle, Medium)
- Object 3: (Red, Square, Medium)

Focusing on attribute 1 (Color), we encounter the values “Red” and “Blue” among these objects. Thus, O_1^S= {Red, Blue}, capturing the distinct color categories present in the cluster. Similarly, O_2^S (for Shape) would be {Circle, Square}, and O_3^S (for Size) would be {Large, Medium}. The cluster S representative $Z_S = (z_1^S, \ldots, z_M^S)$, is then defined by assigning to each attribute m a set of category-frequency pairs:

$$z_m^S = \{(o_{ml}, fS(o_{ml})) \mid o_{ml} \; is \; an \; element \; of \; O_m^S\}.$$

The term $fS(o_{ml})$ denotes the proportional frequency of category o_{ml} in the m-th attribute of cluster S. To compute $fS(o_{ml})$, we count how many objects in S possess the attribute value o_{ml} for attribute m (denoted $\#S(o_{ml})$) and divide that count by p, the total number of objects in the cluster:

$$S(o_{ml}) = \#S(o_{ml}) \; / \, p.$$

In essence, z_m^S is not a single value but a probability-like distribution over the categories of the m-th attribute, reflecting how frequently each category occurs in that cluster.

To determine the similarity between a new object $X = (x_1, \ldots, x_M)$ and the cluster representative Z_S, the k-Representative algorithm uses a simple matching-based dissimilarity measure. For each attribute m, we consider all category values o_{ml} in O_m^S and their frequencies $fS(o_{ml})$. The dissimilarity $dis(X, Z_S)$is defined as:

DOI: 10.14750/ME.2026.038

$$dis(X, Z_S) = \sum_{m=1}^{M} \sum_{o_{ml} \in O_m^S} fS(o_{ml}) \cdot \delta(x_m, o_{ml}) .$$

Here, $\delta(x_m, o_{ml})$ is 0 if $x_m = o_{ml}$, and 1 otherwise. This means the dissimilarity is influenced both by whether x_m matches a commonly occurring category in S (in which case the contribution is low) and by how frequent that category is within the cluster (less common categories influence the sum differently).

To illustrate, returning to our example cluster S and considering a new object: Object 4: (Blue, Circle, Small)

The representative Z_S derived from S would look like this:

- For attribute 1 (Color): {('Red', 0.67), ('Blue', 0.33)}
- For attribute 2 (Shape): {('Circle', 0.67), ('Square', 0.33)}
- For attribute 3 (Size): {('Large', 0.33), ('Medium', 0.67)}

Calculating the dissimilarity step-by-step:
For attribute 1 (Color):

o_{ml} in O_1^S: {('Red'), ('Blue')}

$fS(o_{ml})$: {0.67, 0.33}

δ ('Blue', 'Red') =1, δ ('Blue', 'Blue') =0

Contribution for attribute 1:

$fS('\text{Red}')$ · δ ('Blue', 'Red') + $fS('\text{Blue}')$ · δ ('Blue', 'Blue') = 0.67 · 1 + 0.33 · 0 = 0.67

For attribute 2 (Shape):

o_{ml} in O_2^S: {('Circle'), ('Square')}

$fS(o_{ml})$: {0.67, 0.33}

δ ('Circle', 'Circle') = 0, δ ('Circle', 'Square') =1

Contribution for attribute 2:

$fS('\text{Circle}')$ · δ ('Circle', 'Circle') + $fS('\text{Square}')$ · δ ('Circle', 'Square')= 0.67 · 0 + 0.33 · 1= 0.33

For attribute 3 (Size):

o_{ml} in O_3^S: {('Large'), ('Medium')}

$fS(o_{ml})$: {0.33, 0.67}

δ ('Small', 'Large') = 1, δ ('Small', 'Medium') =1

Contribution for attribute 2:

DOI: 10.14750/ME.2026.038

$fS('\text{Large}') \cdot \delta$ ('Small', 'Large') + $fS('\text{Medium}') \cdot \delta$ ('Small', 'Medium') = 0.33 · 1 + 0.67 · 1 =1

Finally, sum up the contributions from all attributes:

$$dis(Object4, V_S) = 0.67 + 0.33 + 1 = 2$$

Since the total dissimilarity is 2, the cluster assignment of Object 4 depends on comparing this value to the dissimilarities obtained with other cluster representatives. The cluster for which this dissimilarity is minimal is where the object is assigned, indicating the closest categorical "profile."

Through this method, the k-Representative algorithm captures not only the predominant attribute values within a cluster but also their distribution, providing a richer and more flexible characterization of categorical clusters. Unlike methods that rely on a single mode per attribute, k-Representative must manage and update a distribution for each attribute's categories. This can lead to increased computational overhead, particularly for large datasets with many categories. Additionally, the complexity of interpreting frequency distributions may pose challenges in understanding cluster representatives, making it less straightforward for users to interpret cluster meanings.

Numerous specialized extensions have been introduced to address the inherent complexities of clustering categorical data. One noteworthy variant is the k-Centers algorithm [82], which treats cluster centers as probability distributions derived via kernel density estimation. In this approach, indicator vectors and squared Euclidean distance are employed to measure dissimilarities, thereby maintaining the core principles of k-means while effectively accommodating categorical data characteristics.

Beyond k-Centers, additional techniques have emerged, each targeting specific challenges. The fuzzy K-modes algorithm [83] introduces soft assignments, allowing data objects to partially belong to multiple clusters. This flexibility can better capture nuances in complex datasets. Meanwhile, scalable K-modes [84] enhances computational efficiency, making it more practical for large-scale scenarios with vast numbers of objects and attributes. The probabilistic K-modes method [85] integrates probabilistic models to handle uncertainty and variability in categorical attributes, offering a more comprehensive understanding of cluster membership.

These comprehensive efforts reflect ongoing research and innovation to adapt k-means-style algorithms for categorical data analysis. By accommodating categorical attributes through alternative distance measures, frequency-based distributions, or probabilistic techniques, these methods significantly broaden the applicability of clustering algorithms. As a result, they offer effective solutions in diverse areas where categorical data is prevalent, ensuring that k-means and its variants remain integral tools in the data scientist's toolkit**.** In line with these developments, our approach aims to integrate the strengths of conceptual clustering methods with the structural insights of FCA, thus enabling more effective clustering and generalization of categorical concepts within complex lattice structures.

5.3. The Proposed Methods

5.3.1. K-means Dijkstra on Lattice (KDL)

The K-means Dijkstra on Lattice (KDL) approach introduces a form of conceptual clustering tailored to categorical data, integrating FCA with a modified Dijkstra algorithm.

DOI: 10.14750/ME.2026.038

By leveraging the hierarchical structure of the concept lattice, an intrinsic outcome of FCA-KDL advances beyond traditional clustering methods, ensuring that semantic relationships and conceptual hierarchies guide the clustering process. The core phases and principles of the KDL methodology are as follows:

- Data Conversion to Formal Context: The initial step involves converting the categorical dataset into a formal context. This is achieved by representing the data as a binary incidence matrix, where each row corresponds to a distinct object and each column represents an attribute. An entry of '1' in the matrix indicates that the object in that row possesses the attribute denoted by that column, whereas a '0' signifies the absence of that attribute. This binary representation serves as the foundational structure upon which Formal Concept Analysis is applied.
- Formal Concept Derivation**:** Once the formal context is defined, Formal Concept Analysis identifies all the possible formal concepts, each capturing significant relationships among objects and attributes. These concepts form a hierarchical lattice structure that reveals the underlying data organization. Although the number of concepts can grow rapidly, analytical approximations [28], [103] provide insights into this growth, considering both the number of objects, attributes, and the overall size of the context.
- Assigning Edge Weights: At this stage, a directional cost framework is introduced to model the traversal between interconnected concepts within the lattice. By assigning higher costs to certain transitions, such as moving from a parent concept down to its children, this approach can emphasize the significance of particular hierarchical moves. For instance, a downward step might carry a cost of 2, while an upward step might only cost 1. These weighted relationships ensure that the clustering algorithm accurately reflects the relative importance and complexity of moving through different regions of the concept lattice.
- Utilizing Dijkstra's Algorithm for Distance Computation: To evaluate the conceptual distances within the lattice, the method integrates a modified Dijkstra's algorithm. Given the assigned edge weights, Dijkstra's algorithm identifies the shortest path and its associated minimum cost between any two formal concepts. This ensures that the chosen distance metric is sensitive to the lattice's structure, allowing the clustering process to respect and leverage the inherent hierarchical relationships when determining conceptual similarity.
- Deriving and Refining Cluster Centroids (kernels): Once distances within the lattice are established, cluster centroids, elected formal concepts that best represent each cluster are determined. These centroids undergo iterative refinement, with each update recalculating which formal concept minimizes the total distance to all other concepts in the cluster. This iterative process continues until the centroids converge, ensuring that each cluster center is optimally aligned with the inherent structure and relationships in the concept lattice.

The proposed clustering approach, which integrates FCA and Dijkstra's algorithm, leverages a key property of concept lattices: for any two concepts c_1 and c_2 in a concept lattice, there is always at least one path connecting them. Since the lattice is constructed from all possible formal concepts and their hierarchical interrelations, each concept is reachable from any other through a sequence of edges. This ensures that the lattice forms a connected structure, allowing continuous traversal from one concept to another.

DOI: 10.14750/ME.2026.038

To illustrate this, consider two concepts c_1 and c_2. If they share a direct connection ($c_1 \leq c_2$ or $c_2 \leq c_1$), a path between them already exists. If not, we look at the sets of concepts $R(c_1)$ and $R(c_2)$ that are reachable from $c_1$1 and c_2, respectively. If these sets intersect, then there is at least one concept c in the intersection, guaranteeing a path $c_1 \rightarrow c \rightarrow c_1$. If no immediate intersection is found, the search can be extended iteratively by exploring additional reachable concepts until a common one is identified.

This pervasive connectivity is central to the clustering process. Since every concept pair in the lattice is connected, it becomes feasible to compute the least-cost shortest path between any two concepts using the Dijkstra-based distance measure. This, in turn, enables precise cluster formation: each cluster's centroid is identified through concepts that minimize intra-cluster distances, and the inherent lattice structure ensures that these computations are both meaningful and efficient. By exploiting the lattice's connectivity, the proposed method can effectively handle categorical data, respect the conceptual hierarchy, and produce coherent, high-quality clusters.

5.3.1.1. Cluster Centers (Kernel Concepts)

Defining cluster centers, or centroids, within a concept lattice is crucial for effectively applying the K-means Dijkstra on Lattice (KDL) method. These centroids must themselves be formal concepts from the lattice. Their selection and iterative refinement play a key role in minimizing the overall clustering cost. Consider a cluster S composed of formal concepts $\{c_i, \dots, c_{|S|}\}$ where $i = 1,2,\dots,|S|$. The chosen centroid Z is the concept within S that yields the smallest total distance to every other concept in S. Formally:

$$Z = argmin_{Z \in S} \left(\sum_{i=1}^{|S|} d(c_i, Z) \right).$$

Here, $d(c_i, Z)$ represents the Dijkstra-based distance from each concept c_i in the cluster S to a candidate centroid Z. The argmin operator identifies the representative formal concept Z in S that achieves the minimal sum of distances to all other cluster members. Since Z must be a member of S, this approach ensures an efficient search for the optimal centroid.

The existence of such a centroid is guaranteed by the properties of the Dijkstra-based distance measure, making the method generally applicable, regardless of the set of formal concepts at hand. By defining cluster centers as formal concepts, the approach provides both mathematical rigor and practical utility. This strategy enhances the interpretability of clustering results by selecting representative formal concepts for each cluster, ultimately supporting a more comprehensive and insightful analysis of complex concept lattices.

5.3.1.2. The Clustering Algorithm

The K-Means Dijkstra on Lattice (KDL) clustering approach, anchored in FCA and the Dijkstra-based distance framework, operates through a systematic procedure that iteratively refines cluster assignments and identifies optimal centroids rooted in the lattice's conceptual structure.

DOI: 10.14750/ME.2026.038

Algorithm 5.1: K-Means Dijkstra on Lattice (KDL) clustering algorithm

Inputs: k, the number of clusters; $\mathcal{B}$, the lattice of formal concepts.
Output: The resulting clusters $\{S_1, S_2, \ldots, S_k\}$.
Initialize:
Select k formal concepts $\{c_1, c_2, \ldots, c_k\}$ from the lattice $\mathcal{B}$ randomly as the initial centroids of the k clusters.
Assignment:
For each formal concept $c \in \mathcal{B}$ *do*:
Assign c to the cluster S_i for which the Dijkstra-based distance measure $d(c, Z_i)$ is minimized, where Z_i is the centroid of cluster S_i.
Centroid Update:
For each cluster S_i *do*:
Recalculate the centroid Z_i as the formal concept c that minimizes the total distance to all other concepts within S_i
Iteration:
While centroids change between iterations *do*:
Repeat steps 2 and 3.
Finalization:
Output the resulting clusters $\{S_1, S_2, \ldots, S_k\}$.

5.3.1.3. Cost Analysis of KDL Method

This section provides an evaluation of the computational complexity associated with the KDL method, examining each phase from initial cluster setup to the final cluster assignments. Understanding this complexity offers valuable insights into the method's efficiency and scalability.

Let:

- K denote the number of clusters,
- N the number of objects,
- A the number of attributes,
- C the number of concepts,
- E the number of edges in the lattice, and
- B the maximum number of border elements (peripheral concepts with minimal or maximal extent/intent) considered during lattice construction.

Initially, the KDL procedure transforms the categorical dataset into a formal context, a step that involves a binary conversion of each data entry. This preprocessing yields a complexity of $O(NA)$. Following this, the lattice is constructed from the derived formal concepts, requiring operations over all border elements for each concept, resulting in a worst-case complexity of $O(CB)$.

The final stage involves a K-means-like clustering over the lattice-derived concepts. Here, the main computational burden arises from repeatedly determining shortest paths between concept pairs to update cluster assignments and recalibrate centroids. By employing Dijkstra's algorithm and assuming I iterations until convergence, this portion contributes $O(IKC(E + C\ log\ C))$ to the complexity.

Combining these components, the overall time complexity can be approximated as $O(NA + CB + IKC(E + C\ log\ C))$. While this is a rough estimation and may vary based on data characteristics and distributions, focusing on the dominant term for large-scale scenarios simplifies the complexity to $O(IKC(E + C\ log\ C))$.

DOI: 10.14750/ME.2026.038

In summary, the KDL method's complexity grows primarily with the number of concepts and edges in the lattice. Understanding this dependency is essential for selecting suitable parameter values and optimizations to achieve efficient performance in practical clustering scenarios.

5.3.2. K-Means Vector on Lattice (KVL)

This method provides a systematic way to represent categorical data, originally structured as formal concepts, within a numerical framework amenable to conventional clustering techniques. Instead of dealing directly with categorical relationships, KVL transforms each formal concept into a corresponding "concept description vector." In this vectorization step, each concept, however abstract or categorical is represented by a real-valued vector, where each dimension corresponds to a particular attribute. The magnitude of the value in each dimension reflects the attribute's prevalence or significance within that concept.

Once these concept description vectors are obtained, the classical k-means algorithm can be applied directly. By treating each vector as a point in a continuous, high-dimensional space, the standard distance measures and iterative refinement steps of k-means become applicable. Through this process, the concept description vectors are partitioned into k clusters, with each cluster identified by a centroid vector. Vectors within a cluster share a closer similarity to this centroid than to those in other clusters. Consequently, the KVL approach enables the aggregation of related concepts, simplifies the intricate structure of the original categorical data, and facilitates more intuitive, scalable, and numerically-driven cluster analysis.

Definition 4.1 (Concept Description Vector):

Consider a formal concept $c = (X, Y)$, where $X \subseteq G,\ Y \subseteq M$, and the given context $T = (G, M, I)$ comprises a set of objects G and a set of attributes M, with $|\ M\ | = q$ and $|\ G\ | = r$. The incidence relation $I \subseteq G \times M$ is represented by a binary matrix of dimensions $r \times q$, where each entry in the matrix corresponds to whether an attribute is associated with an object (1 if true, 0 if false). Labeling the rows by g_1, g_2,…, g_r and the columns by m_1, m_2,…, m_q, this matrix provides the foundational structure linking objects and attributes. The matrix can be defined as shown in Table 5.1.

Table 5.1. Matrix Corresponding to The Relation I

Objects/Attributes	m_1	m_2	…	m_q
g_1	$I(g_1, m_1)$	$I(g_1, m_2)$	…	$I(g_1, m_q)$
g_2	$I(g_2, m_1)$	$I(g_2, m_2)$	…	$I(g_2, m_q)$
…	…	…	…	…
g_r	$I(g_r, m_1)$	$I(g_r, m_2)$	…	$I(g_r, m_q)$

A concept description vector $c_Y = (v_{m_1}, v_{m_2}, \ldots, v_{m_q})$ captures the essence of the concept c. For each attribute $m_h \in$ M, the component v_{m_h} is computed as follows:

DOI: 10.14750/ME.2026.038

$$v_{m_h} = \begin{cases} 1 & if\ m_h \in B, \\ \frac{1}{r}\sum_{j=1}^{r} I(g_j, m_h) & if\ m_h \notin B, \forall\ g_j \in G, \end{cases}$$

This definition distinguishes between attributes that form part of the concept's intent Y and those that do not. Attributes in the intent are assigned a value of 1, reflecting their strong defining role. Attributes not in the intent are assigned a value corresponding to their average occurrence across all objects G. This frequency-based weighting provides a measure of the attribute's general relevance within the dataset. By constructing the concept description vector in this manner, each vector component encodes how intrinsic an attribute is to the concept. The resulting vector not only supports direct comparisons between concepts but also enables the application of classical numerical clustering techniques, paving the way for more flexible and insightful data analysis.

After constructing the concept description vectors, the KVL approach introduces a concept similarity measure (CS), to evaluate how closely concepts relate to one another. Following Definition 6, Concept Similarity is derived using the Euclidean distance between any two concept description vectors V_{c1} and V_{c2}. This measurement quantifies the proximity of two concepts by considering each corresponding element of their vectors.

Definition 4.1. Concept Similarity (CS):
Let

$$V_{c_1} = \left(V_{c_1 m_1}, V_{c_1 m_2}, \ldots, V_{c_1 m_q}\right).$$

and

$$V_{c_2} = \left(V_{c_2 m_1}, V_{c_2 m_2}, \ldots, V_{c_2 m_q}\right).$$

be the concept description vectors of two distinct concepts c_1 and c_2. The Euclidean distance, which serves as the basis for CS, is given by:

$$CS(V_{c_1}, V_{c_2}) = \sqrt{\left(V_{c_1 m_1} - V_{c_2 m_1}\right)^2 + \left(V_{c_1 m_2} - V_{c_2 m_2}\right)^2 + \ldots + \left(V_{c_1 m_q} - V_{c_2 m_q}\right)^2}.$$

Armed with the concept description vectors and the associated similarity measure, we can apply the classical k-means clustering algorithm. In this process, each concept description vector is treated as a data point in a q-dimensional space. The algorithm groups these vectors into k clusters such that concepts within the same cluster share greater similarity than those in different clusters. Each cluster has a centroid Z_i, defined as the mean of all concept description vectors assigned to that cluster:

$$Z_i = \frac{1}{|S_i|}\sum_{j=1}^{|S_i|} V_{Y_j}, V_{Y_j} \in S_i.$$

DOI: 10.14750/ME.2026.038

where S_i is the set of concept description vectors in the i-th cluster.

The objective of k-means is to minimize the within-cluster sum of squared distances (WCSS) from each concept description vector to its corresponding centroid:

$$\mathrm{Q} = \sum_{i=1}^{k} \sum_{j=1}^{|S_i|} ||V_{Y_j} - Z_i||^2,$$

where:

- S_i is the set of concept description vectors assigned to the *i-th* cluster,
- Z_i is the centroid of cluster i, defined as the mean of all vectors in S_i, and
- $\|\cdot\|$ denotes the Euclidean norm.

By repeatedly assigning vectors to their nearest centroids (based on the CS measure) and then recalculating the centroids, the algorithm proceeds until it converges to a stable configuration, thereby optimally partitioning the concept vectors into coherent, meaningful clusters.

5.3.3. Clustering Algorithm

The clustering procedure unfolds as follows. Consider a formal context $T = (G, M, I)$ and let $V(T)$ represent the set of all derived concept description vectors. Suppose we aim to form K clusters. Initially, randomly select K initial centroids, $Z_t^0 = (A_t, B_t)$ for $(t = 1,2,\ldots K)$, each corresponding to a preliminary cluster $S_t^0 = \{Z_t^0\}$.

Next, assign each concept description vector $v \in V(T)$ to the cluster whose current centroid is nearest to v based on the chosen distance measure. After this initial assignment, recompute each cluster's centroid by taking the average of all vectors assigned to it, thereby updating each cluster center.

This reassignment and centroid calculation process is repeated iteratively. In each iteration, vectors may shift clusters if doing so reduces the overall clustering cost. The process continues until the cluster memberships and their centroids remain stable across consecutive iterations, indicating that the algorithm has converged. The algorithm steps are as follows:

Algorithm 5.2. K-means clustering of concepts

Input: All the description vectors of concepts in $V(T)$, K.
Output: The clusters and corresponding centers.
Initialize:
Set $S_1^i \leftarrow \emptyset$, $S_2^i \leftarrow \emptyset$, ..., $S_k^i \leftarrow \emptyset$;
$i \leftarrow 0$,
Select initial center vectors of K clusters: Z_1^i, Z_2^i,..., Z_k^i;
Assignment:
For each $v \in V(T)$ *do*:
-Find t such that $CS(distance)\left(v, Z_t^i\right) \leq CS(distance)\left(v, Z_j^i\right)$, $(j = 1,2, \ldots, k)$ then,
$v \in S_t^i$;
EndFor
Centroid Update:

DOI: 10.14750/ME.2026.038

For each S_t^i *do*:
$Z_t^{i+1} = \frac{1}{|S_t|}\sum_{s=1}^{|S_t|} v_s\,, v_s \in S_t$
$S_t^{i+1}=\{v \in V(T)|CS(v, Z_t^{i+1}) \leq CS(v, Z_j^{i+1})\}$
EndFor
Convergence Check:
If $Z_t^i=Z_t^{i+1}$, $S_t^i=S_t^{i+1}$, $t = 1,2,\ldots,K$, *then*
Go to "Stop and output the clusters".
Else:
$i = i + 1$,
Go to "Repeat the assignment step".
Output: clusters S_1^i, S_2^i, ..., S_k^i and the corresponding centers Z_1^i, Z_2^i, …, Z_k^i.

Once the clustering process is complete and stable clusters are formed, the concept description vectors in each cluster can be mapped back to their corresponding original concepts from the formal context. This backward mapping leverages the initial construction of concept description vectors, ensuring that the clustering results can be interpreted and analyzed in terms of the actual concepts they represent.

Algorithm 5.3: Mapping Description Vectors Back to Original Concepts

Input: The clusters S_1^i, S_2^i, …, S_k^i and the corresponding centers Z_1^i, Z_2^i,…, Z_k^i.
Output: Clusters of original concepts.
Initialize:
For each $t = 1$ to K, set $NS_t = \emptyset$,
Mapping:
For each vector $v \in S_t^i$:
- Retrieve the corresponding original concept c associated with vector v
- Add concept C to NS_t

Output: the new clusters $NS_1, NS_2, \ldots NS_k$, each containing the original concepts.

This approximation and mapping technique enables efficient and interpretable clustering of concepts within a given context, thereby clarifying the intricate relationships and similarities among the different concepts.

5.3.3.1. Cost Analysis of the KVL Method

A complexity assessment of the KVL approach reveals multiple stages influencing overall performance. Initially, the data undergoes preprocessing where each of the N objects with A attributes is represented in binary form, resulting in a complexity of $O(NA)$. Following preprocessing, C formal concepts are generated, and each concept is represented as an A-dimensional vector, incurring $O(AC)$ time.

Once these vectors are created, the algorithm selects K initial centroids randomly from the C concepts, which adds a cost of $O(K)$. Subsequent phases involve iterative refinement: each iteration requires assigning C concepts to their nearest centroid and then updating those centroids, each iteration costing $O(CK)$. With I iterations until convergence, this totals $O(ICK)$.

In the final step, the algorithm maps the resulting concept vectors back to their original formal concepts, contributing another $O(CK)$ in complexity. Combining all these components yields an approximate total complexity of $O(NA + AC + K + ICK + CK)$. Although this is a heuristic estimation, and real-world complexity may vary depending on the data distribution, focusing on the dominant terms simplifies it to $O(IKC)$. This indicates

DOI: 10.14750/ME.2026.038

that the iterative centroid assignment and update phases primarily influence the scalability and runtime efficiency of the KVL method.

5.4. Experimental Results

This section provides empirical evaluations showcasing the effectiveness and scalability of both the Dijkstra-Based Distance Measure and the two proposed clustering approaches: K-means Dijkstra on Lattice (KDL) and K-means Vector on Lattice (KVL). All experiments were conducted on a Mac system featuring an Apple M1 chip and 8GB of RAM, running macOS 13.2.1. This setup ensures a stable and efficient environment for performance assessment, allowing for consistent comparisons and insights into the practical utility of the methods under real-world conditions. The algorithms developed in this thesis were implemented by the author in Python (version 3.11), using NetworkX for graph computations, scikit-learn for clustering, and Matplotlib for visualization. Formal concepts were derived with a tailored NextClosure routine to ensure canonical enumeration, while iPred was used to construct the Hasse diagram and capture inclusion relations among concepts. To support reproducibility and future research, the full source code including clustering methods, distance functions, and experimental scripts is openly available on GitHub, with the repository link provided at the end of each chapter.

5.4.1. Testing and Evaluation of the Dijkstra-Based Distance

The evaluation of the Dijkstra-based distance measure involved a systematic approach to ensure both its reliability and adaptability across various data settings:

1. Random Generation of Formal Contexts:

   Five distinct formal contexts were randomly generated, each differing in size and density. Table 5.2 summarizes their characteristics. Density here represents the ratio of filled entries (1s) to the total possible entries in the binary matrix, essentially reflecting how much information each context encodes about object-attribute relationships. For instance, Formal Context1 (600 objects, 125 attributes) has a density of 0.10, implying a relatively sparse structure where only 10% of entries are 1. Lower density values indicate sparser relationships, while higher densities signify more attributes per object, thus richer conceptual structures.

2. UCI Datasets Transformation:
   Four well-known datasets from the UCI Machine Learning Repository were included in the analysis. Before experimentation, these datasets were transformed into formal contexts as indicated in Table 5.2. The chosen datasets—Balance-Scale, Breast Cancer, Car Evaluation, and Tae were selected for their public availability and the categorical nature of their attributes:

   - Balance-Scale: Instances reflect different tilt states of a balance scale.
   - Breast Cancer: Instances are categorized as benign or malignant.
   - Car Evaluation: Instances fall into four categories (unacc, acc, good, vgood).
   - Tae (Teaching Assistant Evaluation): Represents teaching performance across multiple semesters, with each instance categorized as low, medium, or high.

DOI: 10.14750/ME.2026.038

3. Formal Concept Extraction:
   The NextClosure algorithm [21] was applied to each formal context to derive all possible formal concepts. The number of formal concepts obtained from each context is listed in Table 5.3. This step is crucial for understanding the underlying patterns and hierarchies within the data.

4. Graph Construction:
   To visualize and analyze the relationships among the extracted formal concepts, graphs were built. The Ipred algorithm [16] was employed to optimally arrange these concepts, considering the influence of density on the resulting diagram's complexity. A higher density often leads to more nodes and edges, reflecting a more intricate concept lattice. In contrast, sparser contexts result in fewer concepts and simpler, more manageable lattices, as illustrated in Table 5.4.

Overall, these preparation steps provided a comprehensive testing environment, ensuring that the Dijkstra-based distance measure was evaluated across a range of densities, dataset complexities, and structural scenarios.

Table 5.2. Characteristics of Random and Real-World Formal Contexts.

Formal Contexts	#objects	#attributes	density
Formal Context1	600	125	0.10
Formal Contex2	11000	30	0.10
Formal Context3	1350	120	0.05
Formal Context4	2000	20	0.15
Formal Context5	12000	20	0.23
Balance-Scale	625	20	0.20
Breast Cancer	182	35	0.25
Tae	151	101	0.04
Car Evaluation	1728	21	0.28

Table 5.3. Formal Concepts Generated from the Formal Contexts in Table 5.2.

Formal Contexts	#formal concepts
Formal Context1	29926
Formal Contex2	15117
Formal Context3	9882
Formal Context4	2989
Formal Context5	39931
Balance-Scale	1297
Breast Cancer	2569
Tae	276
Car Evaluation	8001

Table 5.4. Characteristics of the Generated Lattices.

Formal Contexts	#formal concepts	#inclusion relationship between concepts (edges)
Concept lattice1	29926	122839
Concept lattice2	15117	67040
Concept lattice3	9882	36797
Concept lattice4	2989	12175
Concept lattice5	39931	228427
Balance-Scale	1297	4945
Breast Cancer	2569	9513
Tae	276	619
Car Evaluation	8001	38928

DOI: 10.14750/ME.2026.038

The evaluation process employed the Dijkstra-based distance measure on concept lattices derived from five randomly generated formal contexts and four real-world datasets. These formal contexts differed significantly in terms of objects, attributes, and density, while the real-world datasets encompassed varied domains such as balance scale, breast cancer classification, teaching assistant evaluation, and car evaluation. After establishing the contexts and datasets, FCA techniques, specifically the NextClosure algorithm, were applied to extract formal concepts. The number of resulting concepts ranged widely, from as few as 2989 in Formal Context 4 to as many as 39931 in Formal Context 5.

It is important to note that some datasets, including Balance Scale, Breast Cancer, and Car Evaluation, were also used earlier in Table 4.1 but with different preprocessing configurations (e.g., density, attribute filtering, or encoding methods). In particular, the Breast Cancer dataset was restructured in this chapter, which altered the number of objects and attributes, thereby affecting the number of derived concepts and the resulting lattice structure.

Each set of formal concepts was then represented as a concept lattice constructed via the Ipred algorithm, highlighting the inclusion relationships among concepts. The complexity and size of each context influenced the lattice structure, reflected in the number of inclusion relationships. For the performance assessment, a subset of concept pairs 25% of the total concepts was randomly selected from each lattice. The shortest paths and their costs were computed using the Dijkstra-based measure across ten independent trials. The analysis recorded both the average runtime and the mean distance, providing insights into the efficiency and scalability of the distance measure under varying conditions.

In Figures 5.1 and 5.2, the Dijkstra-based distance measure was applied to lattices derived from randomly generated formal contexts of varying sizes and densities, where in Figure 5.1 shows that the algorithm's runtime generally increases in larger lattices. More objects and attributes produce a greater number of formal concepts, resulting in a lattice with more nodes and edges hence, more computational effort is required for shortest-path calculations. This relationship holds across the all lattices tested, making it clear that denser or larger lattices pose higher computational demands.

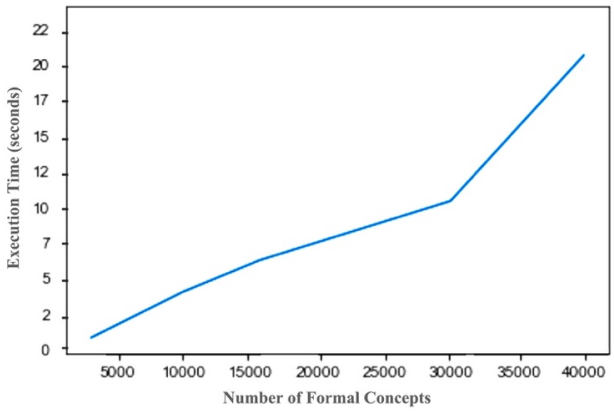


Figure 5.1. Average Runtime vs. Lattice Size for Random Contexts

Figure 5.2 further reveals a consistent pattern when comparing independently generated lattices, often displaying a peak at certain sizes or densities. When a random context produces a moderately sized but fragmented lattice, concept pairs tend to form clusters with relatively few connecting links, increasing shortest-path lengths. In contrast, larger or denser lattices tend to include overlapping attributes that create multiple bridges between clusters, effectively reducing the overall mean distance. These findings highlight how varying

DOI: 10.14750/ME.2026.038

random context parameters generate diverse lattice topologies while still exhibiting predictable trends in both runtime and distance.

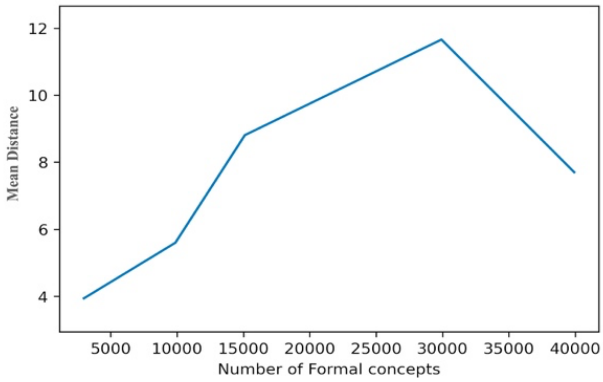


Figure 5.2. Mean Distance vs. Lattice Size for Random Contexts

Figures 5.3 and 5.4 present analogous evaluations of the Car Evaluation, Balance-Scale, Breast Cancer, and Tae datasets, highlighting how real-world categorical data influence both lattice size and concept dispersion. First, Figure 5.3 shows that the runtime scales with the number of concepts; as larger lattices incorporate more nodes and edges, each shortest-path computation requires additional steps, thereby lengthening the average execution time. Second, examining mean distance behaviors as shown in Figure 5.4 reveals that the Car Evaluation dataset despite producing the largest number of formal concepts (8001), exhibits a shorter mean distance than some smaller lattices, suggesting high interconnectivity due to overlapping attributes and more numerous paths between concepts. In contrast, Balance-Scale and Breast Cancer, despite fewer concepts, have higher mean distances, indicating more fragmented lattices with fewer cross-links. As in the random contexts, these real-world datasets can experience peaks in mean distance at certain sizes.

These observations underscore several key insights:

1. Structural Coherence vs. Sheer Size:
   A dataset can generate a large concept lattice yet exhibit relatively short mean distances if its attributes foster dense interconnections among concepts. Conversely, smaller lattices may yield higher average distances when they remain fragmented and lack sufficient bridging attributes.

2. Dynamic Interplay Between Density and Connectivity:
   Across both random and real-world contexts, the layered nature of concept lattices often produces peaks in mean distance. This occurs when there are enough concepts to form loosely connected clusters—rather than fully integrated networks—but not enough overlapping attributes to create extensive cross-links. As additional concepts emerge and overlapping attributes increase, these clusters integrate further, resulting in a decline in the overall mean distance.

3. Robustness of the Dijkstra-Based Approach:
   The Dijkstra-based distance measure consistently captures these subtle structural transitions, underscoring its robustness. Instead of smoothing over inherent differences, it accurately reflects the organization of each dataset, making it a valuable tool for understanding how real-world categorical data are layered or

DOI: 10.14750/ME.2026.038

interlinked. This, in turn, provides deeper insights into the topology and connectivity of concept lattices.

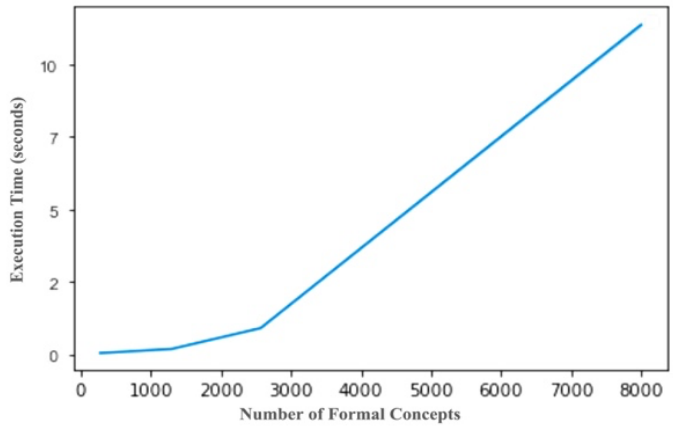


Figure 5.3. Average Runtime vs. Lattice Size for Real-World Datasets

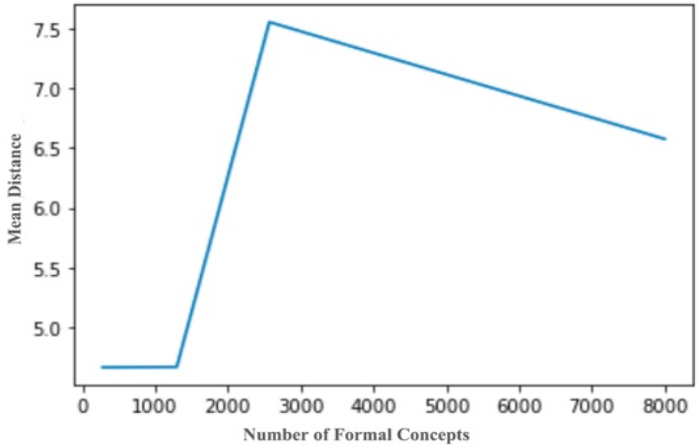


Figure 5.4. Mean Distance vs. Lattice Size for Real-World Datasets

By integrating the Dijkstra-based distance measure with FCA, the approach effectively leverages the hierarchical nature of categorical data. Instead of relying solely on Euclidean metrics, this method interprets dissimilarities as shortest paths within a lattice, thereby more accurately mirroring the relational patterns inherent in categorical datasets. Substituting Euclidean distance with a Dijkstra-based measure in the K-means clustering framework offers a more faithful representation of categorical relationships, enabling more precise cluster identification and analysis. As a result, it provides a powerful, application-agnostic tool for exploring and interpreting complex categorical data, opening up new opportunities for research and practice in data-driven decision-making.

5.4.2. Clustering Performance

To assess clustering quality for categorical data, the Silhouette Coefficient and Davies-Bouldin Index (DBI) are employed. Both measures operate without the need for ground truth labels, making them valuable in practical scenarios.

The Silhouette Coefficient gauges how well each data point fits within its assigned cluster compared to other clusters. Its values range from -1 to 1, where a high positive score

DOI: 10.14750/ME.2026.038

indicates that a point is well-assigned to its cluster, while a negative score suggests a potential misclassification. Formally:

$$\text{Silhouette Score } = (b - a) / max\,(a, b).$$

Here, 'a' represents the average intra-cluster distance (the average distance from a point to other points within the same cluster), and 'b' is the average distance from the point to the points in the nearest neighboring cluster.

In contrast, the Davies-Bouldin Index (DBI) assesses how separated and compact the clusters are. Lower DBI values indicate a more optimal clustering solution. To compute DBI, we proceed as follows:

1. For each cluster S_i, compute the average intra-cluster distance SC_i. This is the average distance of all points in S_i to the cluster's centroid Z_i.
2. Determine the distance d_{ij} between the centroids of each pair of clusters S_i and S_j.
3. For each pair of clusters (i, j), compute the ratio:

$$R_{ij} = (SC_i + SC_j) / d_{ij}.$$

4. For each cluster S_i, identify $R_i = \max(R_{ij})$ across all other clusters SjS_jSj.
5. Finally, the DBI is obtained by averaging all R_i values across the clusters.

Formally:

$$\text{DBI} = \left(\frac{1}{S}\right) \sum R_i.$$

where: S is the total number of clusters

A lower DBI score means clusters are more compact internally and better separated from each other. Both the Silhouette Coefficient and DBI thus provide complementary perspectives on the cluster quality, enabling a robust evaluation of clustering performance in complex categorical data scenarios without requiring predefined labels.

By examining four real-world datasets, as detailed in Table 5.3 and reflected in both the numerical results (Tables 5.5 and 5.6) and graphical trends (Figure. 5.5 and Figure. 5.6), the number of clusters was set to align with each dataset's inherent classes. Averaging the performance across 100 runs per method provided clear insights into how K-means Dijkstra on Lattice (KDL) compares to K-means Vector on Lattice (KVL) in practical clustering scenarios.

Table 5.5. Silhouette Coefficient Outcomes for KDL and KVL Across Various Datasets.

Datasets	KDL	KVL	#Clusters
Balance-Scale	**0.406**	0.128	3
Breast Cancer	**0.239**	0.090	2
Tae	**0.300**	0.092	3
Car Evaluation	**0.563**	0.106	4

Examining the Silhouette Coefficient (Table 5.5) clearly shows that KDL, which inherently respects the lattice graph structure derived from categorical data, consistently surpasses KVL. This advantage is further substantiated by the DBI results (Table 5.6), where

DOI: 10.14750/ME.2026.038

KDL again exhibits superior clustering quality by achieving lower index values across all datasets.

Table 5.6. DBI Results for KDL and KVL Across Different Datasets.

Datasets	KDL	KVL	# Clusters
Balance-Scale	**1.48**	2,64	3
Breast Cancer	**1.83**	2,78	2
Tae	**1.49**	2.62	3
Car Evaluation	**1.90**	2.92	4

The core strength of KDL stems from its integration of FCA and Dijkstra's algorithm. FCA constructs a concept hierarchy reflecting the nuanced relationships in categorical data, while Dijkstra's algorithm finds optimal paths within this hierarchy. By employing a distance measure based on the shortest path between formal concepts, KDL captures the underlying data structure more accurately. This results in more coherent and meaningful clusters.

In contrast, the KVL method, despite simplifying the process by converting categorical data into numerical vectors, may lose critical hierarchical information. Such abstraction can lead to less effective clustering outcomes, as evidenced by higher DBI values and less favorable Silhouette scores.

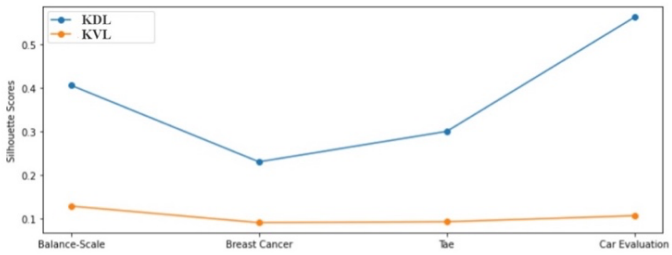


Figure 5.5. Silhouette Scores by Dataset and Method

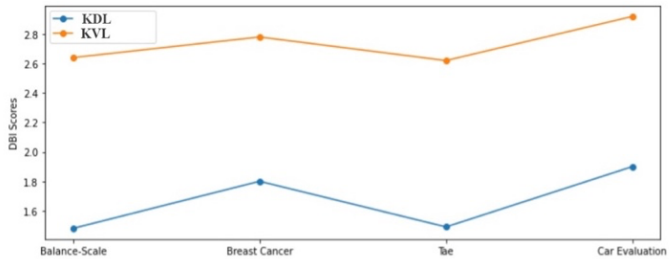


Figure 5.6. DBI Scores by Dataset and Method

Together, these findings underscore the importance of leveraging the inherent structure in categorical datasets. Although KVL remains a viable approach for certain scenarios, the results strongly advocate for methods like KDL especially when the goal is to preserve and utilize the complex relationships implicit in categorical data. In essence, choosing between KDL and KVL should hinge on data characteristics and analytical goals, ensuring that the method aligns with the intrinsic nature of the data at hand.

DOI: 10.14750/ME.2026.038

5.4.3. Scalability Test Results Analysis

Exploring how both K-means Dijkstra on Lattice (KDL) and K-means Vector on Lattice (KVL) respond to changes in the number of clusters provides valuable insights into their scalability. All results reported are based on the average runtime from five independent runs, ensuring the reliability of the performance assessment.

In this experiment, the number of clusters was varied from 2 to 18, while maintaining a constant dataset size. Using the Car Evaluation dataset with 8001 formal concepts as a benchmark, the KVL method displayed a near-linear increase in execution time, as illustrated in Figure. 5.7. The runtime ranged roughly between 44.48 and 51.56 seconds, indicating that KVL scales efficiently with an increasing number of clusters.

In contrast, Figure. 5.8 shows that the KDL method exhibited a steep rise in execution time as the cluster count grew, escalating from about 1926.77 seconds for 2 clusters to approximately 49600.10 seconds for 18 clusters. This substantial jump reflects the computational complexity introduced by navigating the rich lattice structure and multiple concept relationships inherent in KDL.

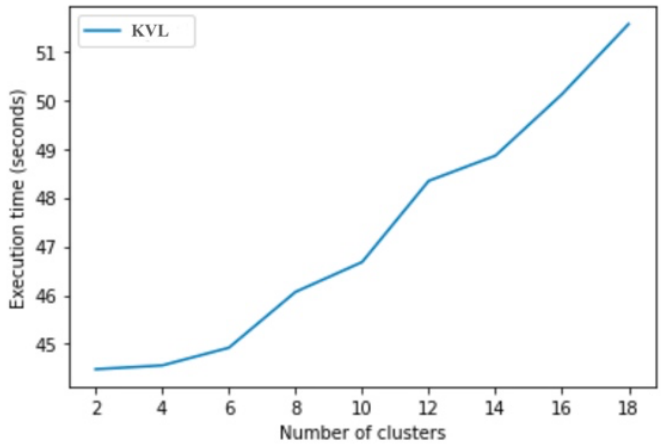


Figure 5.7. KVL Scalability vs. Cluster Count (Car Evaluation Dataset with 8001 Concepts)

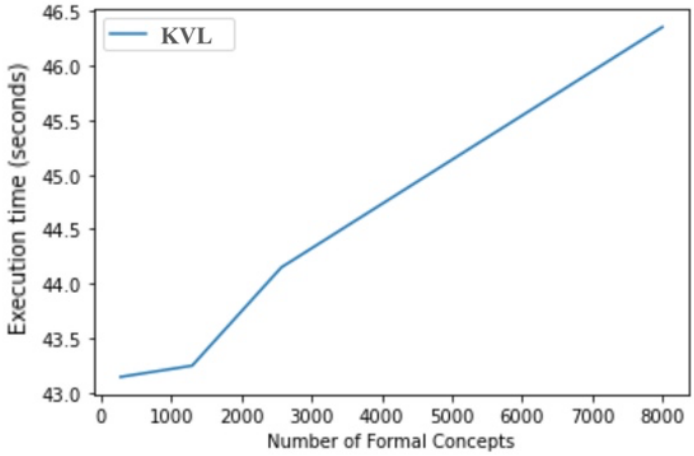


Figure 5.8. KVL Scalability with an Increasing Number of Formal Concepts

These observations suggest that while KVL offers more favorable scalability and computational efficiency with increasing cluster counts, KDL provides more nuanced conceptual results. Ultimately, the choice between methods depends on the analytical

DOI: 10.14750/ME.2026.038

requirements and resource constraints, highlighting a trade-off between scalability and the depth of conceptual structure captured in the clustering process.

5.4.4. Scalability in Relation to the Number of Formal Concepts

When examining scalability with respect to the number of formal concepts, both KDL and KVL methods were tested under a fixed number of clusters (three) across various real-world datasets: Balance-Scale, Breast Cancer, Tae, and Car Evaluation. As illustrated in Figure. 5.9 and Figure. 5.10, the execution times for KVL and KDL were recorded for datasets containing 276, 1297, 2569, and 8001 formal concepts.

Figure. 5.10 reveals that KVL maintains relatively stable execution times as the number of formal concepts grows, demonstrating impressive scalability. The average runtimes remain within a narrow range (43.14 to 46.35 seconds), indicating that KVL efficiently manages increasingly large datasets without substantial performance degradation.

By contrast, Figure. 5.10 shows that KDL experiences a dramatic increase in runtime as the number of formal concepts expands. The execution times escalate from 53.67 seconds to over 2000 seconds, reflecting a substantial computational burden when handling large, complex lattices. Although KDL may offer higher-quality conceptual clustering due to its richer representation, this comes at the cost of reduced scalability.

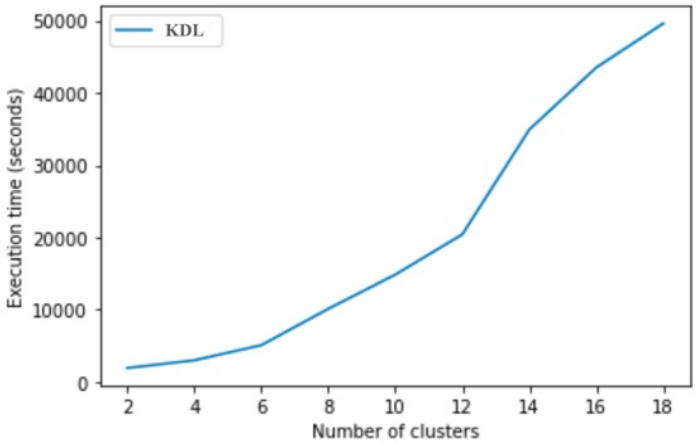


Figure 5.9. KDL Scalability with Increasing Number of Clusters

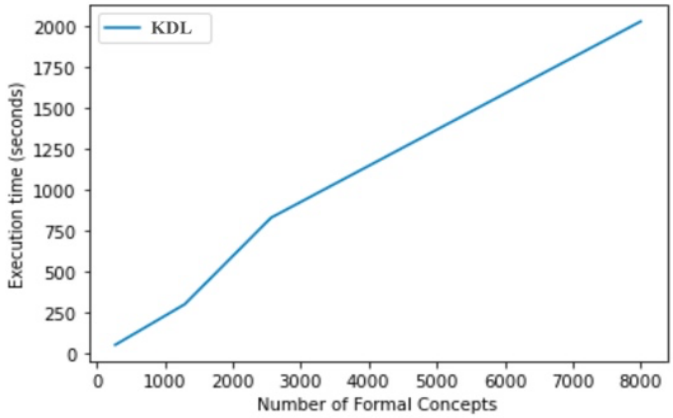


Figure 5.10. KDL Scalability with Increasing Number of Formal Concepts

DOI: 10.14750/ME.2026.038

In summary, while KDL potentially delivers more nuanced clustering results, it is significantly more resource-intensive, limiting its scalability. Conversely, KVL, though possibly less conceptually rich, proves to be far more scalable for larger and more complex datasets. The choice between these methods depends on the priorities and constraints of a given application. Future research could investigate strategies to combine the strengths of both approaches, striving for a method that balances conceptual depth with computational efficiency.

5.5. Summary

Our investigation was guided by the overarching goal of leveraging FCA within conceptual clustering frameworks to effectively reduce and manage the complexity of concept lattices. The introduction of a Dijkstra-based distance measure was pivotal, offering enhanced capability to capture hierarchical relationships in categorical data and reveal deeper structural insights into concept lattices.

We evaluated two clustering methods tailored for FCA contexts: K-means Dijkstra on Lattice (KDL) and K-means Vector on Lattice (KVL). KDL, utilizing the Dijkstra-based distance measure directly on the lattice structure, yielded conceptually rich clusters that preserved inherent hierarchies. However, its scalability diminished as the lattice size grew. In contrast, KVL demonstrated superior scalability but risked oversimplifying hierarchical nuances by converting categorical data into numerical vectors.

Significantly, these methods show promise not only in conceptual clustering but also as reduction tools for FCA concept lattices. Experimental results indicate that KDL and KVL can achieve effective centroid selection from the FCA concept set. Although the execution cost can surpass that of conventional FCA reduction algorithms, these methods still represent viable approaches for concept lattice reduction, particularly when conceptual fidelity or scaling requirements demand careful trade-offs.

Looking ahead, refining these approaches could strike a better balance between conceptual rigor and scalability. Potential avenues include simplifying the lattice construction for KDL or integrating more nuanced distance measures into KVL to preserve categorical hierarchies. Moreover, further integrating the Dijkstra-based measure into k-means could extend their applicability. Ultimately, these findings form a solid foundation for developing clustering methodologies that maintain conceptual depth while efficiently managing lattice complexity, moving closer to more scalable and conceptually sound solutions in FCA-based applications.

Github: *https://github.com/Mdaash/FCA-KVL-and-KDL-/tree/master/KDL_KVL_Methods*

Publications : P_1, P_4

DOI: 10.14750/ME.2026.038

Chapter 6: Kernel Concepts Selection for Efficient Lattice Reduction

6.1. Introduction

FCA provides a powerful framework for conceptualization, its derived concept lattices can become unwieldy, limiting both scalability and insight. Traditional approaches to simplifying these lattices, be they the removal of redundant elements, structural simplifications, or selective filtering, can still struggle to accommodate the dynamic, complex nature of many real-world datasets.

This chapter introduces the Kernel Concept Set Approach (KCS), a novel selection-based methodology designed to address these challenges by integrating concept frequency with a derivation cost function. Unlike typical methods focusing solely on frequency or attribute relevance, KCS offers a flexible cost framework that accounts for both conceptual usage and internal structure. By highlighting specific concepts as cluster centers, the approach not only supports efficient clustering in a general metric space but also preserves core structural insights. Importantly, KCS can serve as a specialized clustering technique, especially within FCA contexts, bringing substantial benefits in interpretability, reduced computational expense, and flexible distance interpretation.

6.2. Kernel Concept Set Approach

The Kernel Concept Set (KCS) method addresses the inherent complexity of concept lattices in FCA, particularly when managing extensive lattices where conventional techniques, such as removing arbitrary elements or selecting objects ad hoc, may overlook critical structures. KCS focuses on two core attributes of each concept: its frequency and the cost of deriving one concept from another. Frequency gauges a concept's prevalence and importance in the dataset, while the derivation cost assesses the effort required to navigate between concepts in the lattice.

Central to KCS is the idea of identifying "kernel concepts," high-frequency concepts strategically positioned in the lattice. By singling out these pivotal elements, KCS preserves both structural coherence and meaningful data relationships during lattice simplification. Furthermore, KCS employs a flexible derivation cost function to measure similarity, thereby accommodating both the real-world usage level of concepts and their internal structure. This dual perspective enriches analysis by spotlighting concept clusters and pinpointing the most essential information within the lattice.

In addition, KCS treats kernel concepts as cluster centroids, making it a powerful clustering approach for formal concepts. This strategy operates in a general metric space, avoiding the need for a vector space, and can yield cost savings relative to typical agglomerative methods. Crucially, KCS not only isolates cluster members but also designates central concepts as cluster representatives, highlighting the lattice's crucial "backbone." Consequently, the KCS method offers a balanced, efficient means to reduce and interpret large FCA lattices while protecting the most valuable insights embedded in the data.

DOI: 10.14750/ME.2026.038

Definition 5.1 (Extended Concept Lattice):

Building upon the standard concept lattice model described in Definition 2.5, the Extended Concept Lattice introduces additional elements to enrich FCA. Specifically, this extension incorporates two core components:

- *A* Frequency Value function, reflecting how often each concept appears or how central it is within the dataset.
- A Derivation Cost function, quantifying the cost or complexity of reaching one concept from another within the lattice's structure.

Definition 5.2 (Frequency Value Function):

Let C represent the set of formal concepts in the extended lattice. A function

$$f: C \rightarrow \mathbb{R}^{+}.$$

assigns a positive real value to each concept $c \in C$. The value $f(c)$ gauges the relative frequency or importance of concept c within the domain.

Definition 5.3 (Derivation Cost Function):

A second function

$$d: C \times C \rightarrow \mathbb{R}^{+} \cup \{0\}.$$

captures the cost of deriving one concept from another. Here, $d(c_1, c_2)$ reflects how much "effort" or "distance" it takes to move from concept c_1 to concept c_2 in the lattice.

- Self-Cost:

$$d\,(c, c) = 0.$$

  No cost is incurred when deriving a concept from itself.

- Asymmetry:

$$d(c_1, c_2) \neq d\,(c_2, c_1).$$

The cost may differ depending on the direction of traversal, mirroring the lattice's hierarchy.

- Integration with a Dijkstra-Based Distance:
  For more refined asymmetrical costs, the lattice edges are weighted so that upward transitions (from child to parent) differ in cost from downward transitions (parent to child). For instance, an upward move might have weight 2, while a downward move might have weight 1. This scheme harnesses a Dijkstra-based shortest path method to capture directional complexities accurately [46].

Definition 5.4 (Distance from a Subset):

DOI: 10.14750/ME.2026.038

For a subset of concepts $K_s \subseteq C$, the distance

$$d(K_s, c) = \min \{d(c_i, c) \mid c_i \in K_s\}.$$

denotes the minimal derivation cost from any concept in K_s to a particular concept c. This measure effectively accounts for the closest "anchor" in K_s when assessing how easily one can reach c.

Definition 5.5 (Frequency-Weighted Derivation Cost):

To incorporate both a concept's significance (its frequency) and the structural cost to reach it, define:

$$d^f(K_s, c) = f(c) \cdot d(K_s, c).$$

This product-based metric balances how often a concept appears with the complexity of accessing it within the lattice.

Definition 5.5 (Kernel Concept Set).

An extended lattice $\mathfrak{B}(d, f, d^f)$ uses these components to identify a *Kernel Concept Set* K_s that satisfies the following:

- Capacity Constraint:

  $|K_s| = S_c$, where S_c is a predefined size limit.

- Optimization Constraint:

  K_s should minimize the cumulative derivation cost across the lattice. Formally:

  $$K_s = argmin_{K_s \subset K}\{\sum_{c \in K} d^f(K_s, c) \mid |K_s| \leq S_c\}.$$

  This enforces an optimal coverage of the concept set using only S_c kernel concepts.

- Role in Lattice Simplification:

  By focusing on these kernel concepts which both appear often (high frequency) and are strategically positioned (low derivation cost) the approach zeroes in on the lattice's structural "backbone." It thereby condenses the lattice into its most informative subset, enhancing manageability and preserving core relationships during analysis.

Overall, these definitions provide a systematic framework for extending an FCA concept lattice with frequency-based prioritization and cost-aware navigation, enabling more powerful reduction, clustering, and insight extraction in complex or large datasets.

DOI: 10.14750/ME.2026.038

6.2.1. Optimized Greedy Algorithm for Determining a Kernel Concept Set

To address the significant computational demands often associated with large concept lattices in FCA, we introduce an Optimized Greedy Algorithm (Algorithm 5.1) that efficiently identifies a Kernel Concept Set (KCS). The algorithm is devised to systematically construct an optimal subset of concepts that minimizes total derivation costs across the lattice. This total cost encapsulates the aggregate effort of deriving every other concept from a chosen kernel set of core concepts. By focusing on such pivotal concepts, the procedure inherently reduces the lattice's size and complexity, enhancing interpretability while retaining essential structural insights.

Algorithm 5.1: Optimized Greedy Algorithm

Input:

- Concept Lattice $\mathfrak{B}$ *(K, ≤)*
- Frequency Value Function $f: C \rightarrow R^+$
- Maximum Core Set Size S_c
- Transition Cost: $upward \leftarrow 2, downward \leftarrow 1$

Output:

- Kernel Concept Set K_s

Algorithm Steps:

1. Initialization:
   - Construct the Concept Lattice $\mathfrak{B}(C, \leq)$.
   - Initialize Kernel Set K_s as an empty set.
   - Assign Frequency Values $f(c)$ to each concept c in the lattice.
2. Ancestors and Descendants Preprocessing:
   - For each concept c in the lattice, identify its ancestors and descendants.
   - Prepare a memoization dictionary to store the minimal derivation costs.
3. Derivation Cost Calculation:
   - For each concept c in the lattice:
     - Use Dijkstra's algorithm to calculate the minimal derivation cost $d\ (K_s, c)$ to every other concept.
     - Store the costs in a structured way for quick retrieval and use memorization to avoid redundant calculations.
4. Core set identification with Sub-Lattice Optimization:
   - Define S_c as the maximum size for the Kernel set.
   - Initialize
     $best\_cost \leftarrow \infty$, *best_candidate* ← None.
   - Iteratively expand K_s:
     - For each candidate concept not in K_s, construct or retrieve a relevant sub-lattice Algorithm 5.2.
     - Calculate the potential reduction in aggregated derivation cost if the candidate were added to K_s.
     - Update $best\_cost$ and *best_candidate* accordingly.
     - Add the *best_candidate* to K_s and update the cost.
     - Continue until $|K_s|=S_c$ or no further reduction in cost is possible.
5. Result Analysis:

Return the final K_s as the kernel concept set that minimizes the aggregated derivation cost while adhering to the size constraint $|K_s|=S_c$.

The computational complexity of the proposed approach is influenced by several factors, most notably the number of concepts within the lattice (denoted by C) and the structure of their interconnections. During the preprocessing stage, identifying ancestors and descendants for each concept in a densely connected lattice can lead to an $O(C^2)$ overhead. A naive derivation cost calculation for all concept pairs, which employs Dijkstra's algorithm,

DOI: 10.14750/ME.2026.038

might appear to scale as $O(C^3)$. However, by confining each cost determination to a focused sub-lattice of average size s, the effective complexity adjusts to roughly $O(C \times s)$. Within the iterative kernel construction step, adding each new kernel concept involves recalculating aggregated derivation costs, but again only on localized sub-lattices and with memoization to avoid repeated computations. This further step is typically bounded by $O(S_c \times s)$ is the maximum permitted size of the kernel set. Consequently, the overall time requirement primarily combines the derivation cost computations $O(C \times s)$ with the iterative kernel set expansions $(S_c \times s)$, yielding a marked reduction in comparison to a more naive global approach.

Algorithmic routines such as sub-lattice construction (presented in Algorithm 5.2) are crucial for reducing the size of the problem space:

1. Defining the Sub-Lattice
   - Identify a compact subset of concepts (and their interconnections) directly relevant to the current calculation.
   - This subset often centers on the target concept(s) and the kernel set members.
2. Selective Inclusion
   - Only nodes (concepts) and edges (relationships) pertinent to the cost evaluation or kernel set update are included, minimizing overhead.
3. Dynamic Construction
   - As the algorithm updates the kernel set or refines potential candidates, sub-lattices are rebuilt or adjusted to ensure accuracy and relevance.
4. Scalability
   - By confining computations to smaller sub-lattices, the method accommodates lattices of larger overall size without incurring prohibitive computational costs.

Algorithm 5.2: Steps for Building a Sub-Lattice

1. Initialize Relevant Concepts:
   - Start with an empty set to hold all relevant concepts.
   - Add the two concepts, A and B, to the relevant concepts set.
2. Add Ancestors and Descendants:
   - Include all ancestors of A into the relevant concepts set.
   - Include all descendants of A into the relevant concepts set.
   - Repeat the process for node B, adding both its ancestors and descendants to the relevant concepts set.
3. Create Sub-Lattice:
   - Initialize an empty dictionary to represent the sub-lattice.
   - For each concept in the relevant concepts set, do the following:
     - Initialize an empty list to store the neighbors of the concept.
     - Retrieve the list of neighbors from the full lattice dictionary.
     - Include a neighbor in the concept's neighbor list only if the neighbor is also in the relevant concepts set.
   - Assign the neighbor list to the concept in the sub-lattice dictionary.
4. Return Sub-Lattice:
   - The sub-lattice containing only the relevant concepts and edges is now constructed.
   - Return the sub-lattice dictionary.

By applying these optimization methods, the algorithm strategically narrows the scope of its computations while still preserving a comprehensive view of the lattice. This balanced approach results in a kernel set that is both cost-effective and representative, exemplifying how depth and breadth can be maintained in the analysis of large and intricate concept lattices.

DOI: 10.14750/ME.2026.038

6.3. Experimental Setup and Methodology

We conducted our algorithm's implementation and evaluation within a Python-based environment, leveraging its widespread community support and broad selection of development utilities. All experiments ran on a Mac equipped with an Apple M1 processor and 8GB of RAM, operating under Mac OS 14.3.1, ensuring a stable and efficient testing platform for diverse computational.

6.3.1. Clustering Performance

A comparative evaluation of the proposed Kernel Concept Set Approach (KCS) against the K-means Dijkstra on Lattice (KDL) [46] method was carried out using four real-world datasets (see Table 4.1). To measure clustering quality without the requirement of labeled data, we relied on the Silhouette Coefficient and the Davies-Bouldin Index (DBI). Across all datasets tested, KCS consistently surpassed KDL, reflecting more coherent within-cluster organization and clearer separation among clusters. Specifically, KCS achieved higher Silhouette Coefficient values, for example, 0.406 and 0.680 on the Balance-Scale and Car Evaluation datasets, respectively, and lower DBI scores (e.g., 1.72 and 1.41), indicative of tighter, well-separated clusters.

These strong outcomes stem from KCS's strategy of choosing kernel concepts as cluster centers based on both concept frequency and derivation cost. By situating clusters around pivotal kernel concepts, KCS effectively captures the essential structure of large lattices, lowering complexity while maintaining meaningful relationships among concepts. Moreover, KCS operates within a general metric space, circumventing the overhead of vector-space transformations, and thus reduces computational costs relative to some traditional approaches. Its capability to identify both cluster memberships and centroids facilitates deeper insights into the data's inherent patterns, ultimately supporting a more efficient, interpretable, and lattice-focused analysis.

The findings summarized in Tables 6.1 and 6.2, along with the corresponding visual representations in Figure. 6.1 and Figure. 6.2, underscore the Kernel Concept Set (KCS) approach's notably stronger clustering performance compared to the K-means Dijkstra on Lattice (KDL) method across multiple datasets. This advantage arises from KCS's distinctive use of the concept lattice's inherent complexity for clustering, thereby offering a more fine-grained and effective analysis of categorical data. In contrast to the KDL method—which capitalizes on the lattice structure and Dijkstra's algorithm—KCS centers on identifying a "kernel" of concepts, prioritizing their frequency and derivation cost. By focusing on a lattice's most meaningful elements, this strategy not only reduces the volume of information needing analysis but also yields higher-quality clusters by designating these kernel concepts as cluster hubs.

Table 6.1. Silhouette Scores Comparing KDL and KCS Methods Across Datasets.

Datasets	KDL	KCS	#Clusters
Balance-Scale	0.275	**0.406**	3
Breast Cancer	0.125	**0.351**	2
Tae	0.163	**0.393**	3
Car Evaluation	0.382	**0.680**	4

Significantly, this quality advantage is reflected in KCS's improved Silhouette Coefficients and lower Davies-Bouldin Index values, indicative of more cohesive within-

DOI: 10.14750/ME.2026.038

cluster relationships and clearer delineations between clusters than those achieved by KDL. Overall, these results affirm that a selective, concept-focused methodology—like that employed by KCS—can substantially elevate clustering outcomes by directly engaging with the most pivotal facets of categorical data and their hierarchical interconnections.

Table 6.2. DBI Index Scores Comparing KDL and KCS Methods Across datasets.

Datasets	KDL	KCS	# Clusters
Balance-Scale	2.67	**1.72**	3
Breast Cancer	2.88	**1.35**	2
Tae	2.12	**1.70**	3
Car Evaluation	3.34	**1.41**	4

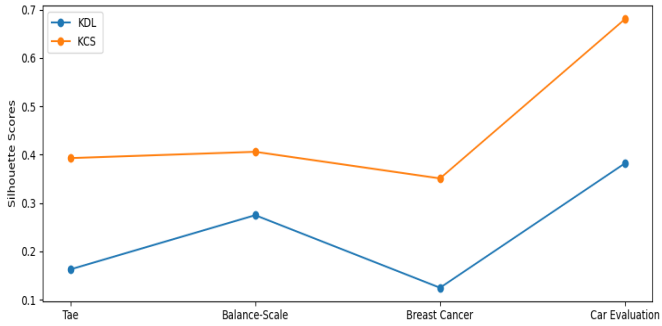


Figure 6.1. Silhouette Scores by Dataset and Method.

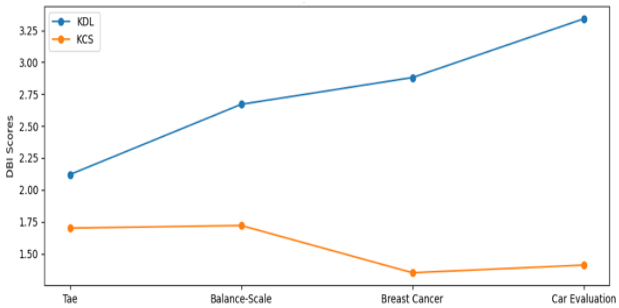


Figure 6.2. DBI Scores by Dataset and Method

6.3.2. Influence of Lattice Size on Runtime

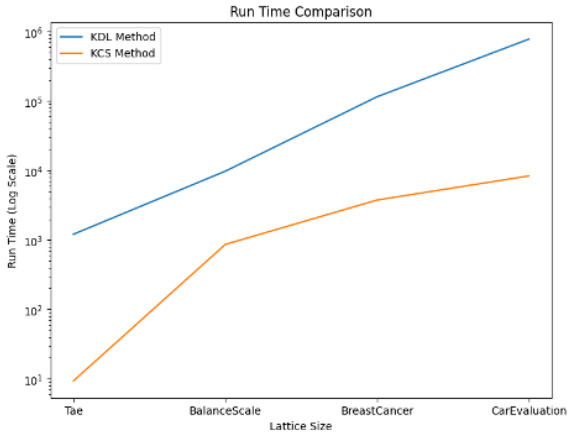
In a targeted experimental comparison, we examined how the Kernel Concept Set Approach (KCS) measures up against the K-means Dijkstra on Lattice (KDL) method within the framework of FCA. By using the datasets outlined in Table 4.1, we investigated how these methods handle increasingly complex lattice structures, focusing on their runtime as the principal performance metric.

As depicted in Figure. 6.3, the tested approaches diverge markedly in efficiency once lattice size grows. Although KDL demonstrated acceptable performance for relatively modest lattices, it exhibited substantial scalability and runtime issues with larger structures. In stark contrast, KCS maintained strong efficiency across the entire range of lattice sizes. For instance, when processing the "Tae" dataset (276 concepts), KDL required 1,210.14 seconds, whereas KCS completed the same task in just 9.35 seconds. A similar pattern

DOI: 10.14750/ME.2026.038

appeared in the "Car Evaluation" dataset, where runtime dropped from 781,799.93 seconds (KDL) to 8,361.93 seconds (KCS) for 3,596 concepts.

These outcomes highlight the KCS method's superior adaptability and computational economy, making it a more effective choice for FCA applications spanning varied and especially larger lattice complexities. The capacity to reduce runtime substantially across different scales underscores the feasibility of using KCS in data-intensive environments. By dramatically lessening the time required to process substantial concept lattices, KCS significantly broadens FCA's practical utility in analyzing complex datasets—ultimately setting a new performance benchmark, as illustrated by the results in Figure. 6.3.



6.3. Comparative Performance Analysis of KCS and KDL Methods Across Diverse Lattice Sizes

6.3.3. Experiment with the Teaching Assistant Evaluation Dataset

A specific demonstration involves the Teaching Assistant Evaluation dataset sourced from the UCI KDD Archive. This dataset captures the performance of 151 teaching assistants (TAs) in the University of Wisconsin-Madison's Statistics Department across various semesters, including both standard academic terms and summer sessions. Publicly available at UCI KDD, the dataset provides a valuable basis for investigations into teaching effectiveness. Six categorical attributes, encompassing elements such as TA language background (English speaker or not), course instructor (25 categories), course type (26 variants), semester format (summer or regular), and class size, collectively enable a multifaceted view of TA performance assessments.

For use within FCA, each categorical attribute is transformed into Boolean form, producing a formal context containing 151 rows (one per TA assignment) and 101 columns (attributes) at a density of 0.05. Table 6.3 illustrates a smaller portion of the data comprising 10 TA assignments and 8 attributes, while Figure 6.4 depicts the resulting concept lattice through a line diagram generated using the ConExp (Concept Explorer) software. This initial demonstration provides a concise view of the relationships and structure revealed by FCA.

When the entire dataset is processed, the Kernel Concept Set (KCS) method proves to be highly effective at consolidating and clarifying the concept lattice's 276 concepts. Applying an initial maximum kernel size S_c of 5% yields a kernel containing 14 key concepts (see Table A.2 of Appendix A), registering a total derivation cost of 30,808. Although the frequency values in these concepts were assigned to illustrate the process, the result already highlights salient patterns in the TA assignments, including a marked emphasis on certain attributes (e.g., semester format or language proficiency). For instance, two of the concepts

DOI: 10.14750/ME.2026.038

alone cover 138 TAs out of 151, indicating a notable preference for non-English-speaking TAs in regular semesters.

Table 6.3. Formal Context about Subset of Tas Dataset.

	Class_Size_17	Eng_Nat_spk_1	Eng_Nat_Spk_2	Summer_or_Regular_1	Summer_or_Regular_2	Course_3	Course_Instructor_13	Course_Instructor_23
TA 1	X	X		X		X		X
TA 2	X	X			X	X		X
TA 3	X		X		X	X	X	
TA 4	X		X	X		X		X
TA 5	X	X			X	X	X	
TA 6	X		X		X	X		X
TA 7	X		X		X	X		X
TA 8	X	X		X		X	X	
TA 9	X	X		X		X	X	
TA 10	X	X		X		X	X	

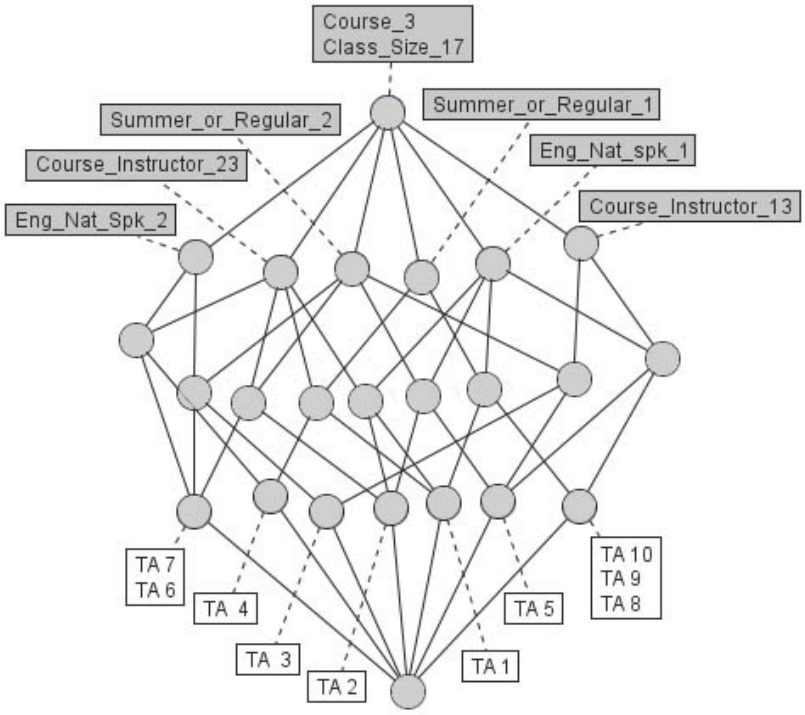


Figure 6.4. Concept Lattice Derived from the Formal Context of Tae Dataset Table 5.4.

Raising S_c to 8% preserves the original 14 concepts while adding eight more, leading to a 22-concept kernel with a reduced total derivation cost of 26,768 (Table A.3 of Appendix A). These newly integrated concepts bring more refined insights, including further details about class sizes, course types, and TA language patterns. Such additions reveal more complex assignment practices, for example, employing non-English-speaking TAs for

DOI: 10.14750/ME.2026.038

specific regular-semester courses and assigning English-speaking TAs in summer sessions. This reconfiguration of the kernel concept set not only uncovers deeper relationships but also underscores the KCS approach's flexibility in uncovering multiple hierarchical layers within the data.

Reducing the aggregate derivation cost from 30,808 to 26,768 underscores how effectively KCS refines the lattice, pinpointing vital concepts that encapsulate the dataset's most pertinent patterns. Moreover, by compressing these key relationships into a concise subset of the full concept set, KCS makes the analysis both more streamlined and more illuminating. This capability is especially valuable for exploring data-intensive educational contexts, where capturing essential interactions, such as instructor preferences or course attributes, is crucial for decision-making.

This efficiency becomes clearer when examining how the kernel set size expands from 5% to 20%, as shown in Figure. 6.5. With a kernel set size S_c initially at 5%, the derivation cost starts at 30,808 and steadily declines as the kernel increases, dropping to 24,274 at 10%, 19,782 at 15%, and reaching 16,132 at 20%. This continuous decrease highlights one of the KCS approach's central strengths: the capacity to include more concepts in the kernel set while keeping the overall complexity in check. The additional concepts fit smoothly into the existing lattice, maintaining a streamlined analytical process even as the dataset coverage grows broader. This well-balanced integration confirms the KCS method's scalability and adaptability for complex data exploration, enabling richer insights and more informed conclusions without placing undue computational strain on the analysis.

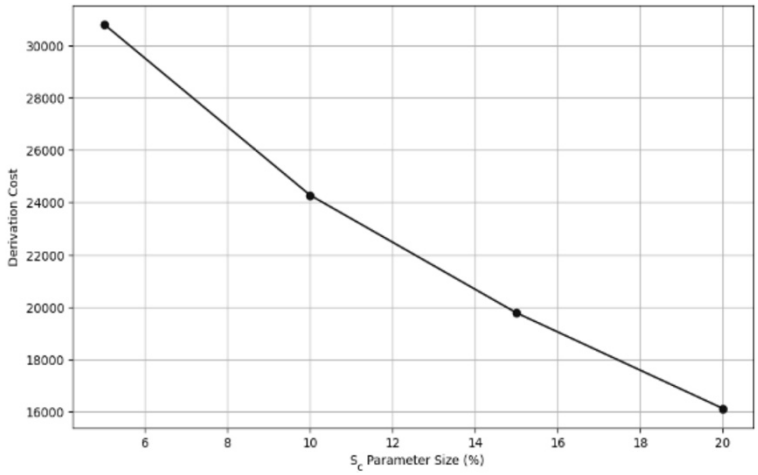


Figure 6.5. Trend of Decreasing Derivation Cost with Incremental Expansion of Kernel Set Size (S_c)

6.4. Summary

This chapter introduces the Kernel Concept Set Approach (KCS) as an innovative extension of FCA, targeting the inherent complexity of large concept lattices. By integrating concept frequency with a flexible derivation cost function, KCS goes beyond conventional frequency- or attribute-based filtering methods. Specifically, KCS strategically pinpoints kernel concepts to serve as pivotal cluster centroids, emphasizing both their prevalence (frequency) and the effort to derive one concept from another (cost).

Compared to the K-means Dijkstra on Lattice (KDL) method, KCS demonstrates superior efficiency and clearer structural insight, even within a general metric space where

DOI: 10.14750/ME.2026.038

many standard approaches incur higher overhead. By highlighting key concepts and reducing derivation costs, KCS preserves the essential lattice structure, yielding high-quality clustering results. Crucially, this approach also enables scalable approximation of formal concept lattices, accommodating larger datasets without sacrificing interpretability or performance. Consequently, KCS stands as a valuable, cost-effective tool for researchers and practitioners seeking deeper insights and more streamlined analysis in FCA-driven data exploration.

Github: https://github.com/Mdaash/KCS_Approach/blob/master/KCS_Method_v1.ipynb

Publications : P_2, P_1, P_4

DOI: 10.14750/ME.2026.038

Chapter 7: Mining Kernel Concepts: A Cost-Optimized Concept Set Generation Method

7.1. Introduction

This chapter introduces a new framework for concept lattice reduction, focusing on an optimal balance between expressive power and computational feasibility. Unlike conventional methods that emphasize frequency filters or attribute-based pruning, our model employs a heuristic and machine learning–assisted strategy to pinpoint a small "kernel" of high-frequency concepts. These selected kernel concepts form a finite memory structure, with a specialized mapping function ensuring each concept is uniquely and transparently represented. The method is further bolstered by a Genetic Algorithm (GA) tasked with optimizing the kernel selection, aiming to minimize a global generation cost while preserving lattice integrity. Extensive tests confirm that our GA-based approach outperforms a benchmark Simulated Annealing method in both speed and scalability. The chapter also demonstrates a linguistic-based cost model for defining kernel vocabularies, showcasing the versatility of our solution for diverse contexts and data domains. Our Main Contributions:

- Development of a Novel Reduction Model: We introduce a mechanism that integrates a derivation cost function with a robust optimization procedure, enabling the construction of a simplified yet expressive concept lattice.
- Genetic Algorithm with Machine Learning Support: A neural network module predicts chromosome segment fitness, generating an efficient starting population for the GA, thus enhancing convergence speed.
- Flexible Probability Distribution for Concept Prioritization: Our system accommodates various probability distributions $P(s)$ across concepts, enabling tailored solutions in domains with different analytical requirements.
- Injective Mapping Function: By ensuring each concept is encoded as a unique word sequence, the mapping function prevents ambiguity and preserves clarity during lattice reduction.

Our approach provides multiple benefits that significantly improve both the scalability and usability of FCA:

- Scalability: Adjustable kernel concept selection through input parameters allows users to generate compact or more expansive concept sets, matching specific data complexity.
- Approximation of Full Lattice: The resulting kernel concepts effectively approximate the entire concept lattice, retaining crucial relational patterns while minimizing overall complexity.
- Enhanced Clarity: The injective mapping function, coupled with the kernel's high-frequency elements, yields a more interpretable representation of concepts.
- Cognitive Alignment: Aligning the reduced structure with linguistic and cognitive principles lowers the mental overhead for understanding and navigating the lattice.

DOI: 10.14750/ME.2026.038

- Adaptability: Configurable memory sets W_M and selection thresholds facilitate broad adaptability across various domain-specific vocabularies and semantic demands.

Against this backdrop, the following sections detail the design of our reduction method, elaborate on the Genetic Algorithm for kernel concept selection, and evaluate the resulting model through comprehensive experiments.

7.2. Proposed Method

To systematically reduce a concept lattice while maintaining both expressiveness and derivational efficiency, we propose selecting a targeted kernel subset of concepts. Guided by the compactness and clarity inherent in human language, our method relies on a finite "memory" of frequently used concepts, applies an injective mapping function to guarantee a unique representation for each concept, and utilizes optimization algorithms focused on minimizing overall generation cost. By aligning with cognitive and linguistic principles, this strategy not only streamlines computational tasks but also enhances the interpretability and practical utility of the resulting lattice.

7.2.1. Kernel Set C_M

We begin by assigning a probability value to every concept in the concept lattice $L = (C, \leq)$, These probabilities form a distribution p: $C \rightarrow [0,1]$ such that

$$\sum\nolimits_{c \in C} p_c = 1.$$

Each probability reflects how frequently a given concept is used. For instance, the concept "bread" is typically used more often than "petrichor." In addition to the concept lattice, this probability distribution serves as an integral part of the input data.

The first step in reducing the concept set relies on probability-based filtering. Specifically, we introduce a probability threshold p_F. Any concept whose probability value is below this threshold is removed from consideration, leaving us with the set of frequent concepts,

$$C_F = \{c \in C \mid p(c) \geq p_F\}.$$

Note that, in general, C_F does not form a lattice. From C_F, we select a finite subset of concepts, known as the kernel C_M,

$$C_M = \{C_{M,1}, C_{M,2}, \ldots, C_{M,D}\} \subset C_F, .$$

where D is the size of the kernel set. This finite size is a key attribute: it is chosen based on the specific requirements of an application and the limitations of available resources, thereby ensuring representations that are both scalable and manageable. Moreover, the kernel set's properties help guarantee its effectiveness and dependability in the model.

DOI: 10.14750/ME.2026.038

The kernel concepts act as special cluster centroids within the target concept set. Clustering, commonly employed in data analysis, reduces data volume such that subsequent analyses can target whole clusters rather than individual items, thereby optimizing resource usage. In particular, conceptual clustering refines standard clustering methods (like k-means or hierarchical agglomerative clustering) to work with semantic concept domains. In this study, we use an evolutionary strategy to optimize the positions of the cluster centers.

One application of this kernel concept model lies in refining linguistic concept representations. In the language model considered here, each kernel concept corresponds to a single word in the available vocabulary, each of these words is a single-word linguistic unit that forms the foundation for representing the broader set of concepts.

7.2.2. Kernel Selection Method

Given a kernel set C_M, we define a cost function h_{C_M}:

$$h_{C_M}: \mathrm{C} \rightarrow \mathbb{R}^+ .$$

where,

$$h_{C_M}(c) = g(\{d(c_k \in C_M, c)\}).$$

where $d(c_k, c)$ represents the cost of deriving a representation of c from c_k, and g is a function applied to the set of these distances. A common choice for g is the min function. The main objective is to identify the kernel that minimizes the overall mapping costs, which is calculated as

$$h(C_M) = \sum\nolimits_{c \in C} p_c \, h_{C_M}(c).$$

Additionally, there is a constraint on the size of the kernel set:

$$|C_M| \leq K.$$

where K is a predefined integer. Minimizing $h(C_M)$ by optimally determining the kernels C_M is the core goal. Through this approach, we significantly enhance FCA by reducing the complexity of the concept lattice via a careful selection of key concepts. This, in turn, supports more efficient knowledge representation and further broadens the potential applications of FCA across various complex domains.

If, in a particular case, $h(C_M)$ is defined as the sum of element-wise costs

$$h(C_M) = \sum\nolimits_{c_k \in C_M} d(c, c_k).$$

and taking the following weight value:

$$w_c = 1,$$

DOI: 10.14750/ME.2026.038

the problem becomes analogous to the well-known knapsack problem. Specifically, if we use an indicator variable x_i to denote whether a concept c_i is part of the kernel, then the cost function can be expressed as:

$$h(C_M) = \sum_{c \in C} p_c \sum_{i \in C} d(c,i)\, x_i = \sum_{i \in C} (d(c,i) \sum_{c \in C} p_c)\; x_i = \sum_i v_i x_i,$$

with a capacity constraint of

$$\sum_{i \in C} w_i x_i \; \leq K.$$

Since the knapsack problem is NP-complete, the general form of our optimization task is at least as challenging. As a result, we employ heuristic methods, namely, a genetic algorithm (GA) and a simulated annealing approach to tackle the problem.

When using the GA, we search efficiently for an approximately optimal subset (kernel) and its associated mapping function, aiming to minimize the total expected cost. Because the concept lattice can be exponentially large, an exact solution is often infeasible; the GA instead balances exploring a broad solution space with systematically refining promising candidates to converge on an effective solution.

Simulated annealing serves as the second baseline in our evaluations. This well-established method is particularly suited to large and complex optimization spaces. Beginning with an initial solution, it iteratively generates new "neighbor" solutions, and whether a new solution is accepted depends on a probabilistic factor governed by a temperature parameter. This mechanism allows the algorithm to escape local optima, potentially leading to further improvements in the solution.

7.2.3. Optimization of the Genetic Algorithm

The Genetic Algorithm (GA) in our framework conducts a chromosome-level evaluation that primarily governs the selection operation, also influencing crossover and mutation. In both the mutation phase and the generation of the initial population, a uniform random selection is typically used, which tends to be less efficient than a fitness-based approach.

Algorithm 7.1: Genetic Algorithm for Optimizing the Memory Subset

1. Input
   - Concept Lattice: L
   - Frequency Distribution: P
   - Kernel Size Constraint: K
2. Output
   - The optimal kernel subset C_M
3. Algorithm

   3.1. Initialize the Kernel
   Begin by setting the kernel subset to C_A, the atomic concepts.
   3.2. Determine Chromosome Length
   Let $L = \mid C \setminus C_A \mid$(each chromosome is indicating a potential memory subset).

DOI: 10.14750/ME.2026.038

3.3. Configure Genetic Algorithm Parameters
$N_{\text{population}}$, $N_{\text{generation}}$, $P_{\text{selection}}$,$P_{crossover}$, $P_{mutation}$
3.4. Genetic Algorithm Loop
3.4.1. Population Initialization
Generate $N_{\text{population}}$chromosomes. Each chromosome with exactly $K - K_A$ ones, ensuring the memory constraint is satisfied.
3.4.2. Evaluate Fitness
For each chromosome (memory subset candidate C_M), compute $h(C_M)$. The fitness is inversely proportional to this cost $h(C_M)$
3.4.3. Selection Operation
Randomly choose chromosomes from the current population according to their fitness (fitter chromosomes have a higher chance of being selected).
3.4.4. Crossover
Apply single-point crossover among the selected chromosomes to produce new offspring.
3.4.5. Mutation
Use bit-flip mutation to invert randomly chosen bits ($0 \rightarrow 1 \; or \; 1 \rightarrow 0$) in the offspring.
3.4.6. Repair Phase
Ensure each offspring still meets the $K - K_A$ ones constraint.
If there are too many ones, flip random ones to zero until the count is correct; if too few, flip random zeros to ones until the required number of ones is reached.
3.4.7. Replacement
Form the new population from the resulting offspring after repair.
3.5. Termination Condition: After $N_{\text{generation}}$ iterations, select the chromosome with the highest fitness as the best solution.
3.6. Return: Output the optimal chromosome, which corresponds to the best-performing kernel subset C_M.

To address this limitation, we introduce a machine learning module for predicting the relevance of any subset of concepts in L. This module can directly propose kernel set candidates without requiring exhaustive enumeration. Our method proceeds as follows:

For any subset of concepts $S \subset C$, we introduce a fitness function

$$f(S) = \frac{\sum_{c' \in C_c} p_c \, h(c', C_M)}{|C_c|},$$

where C_c is defined as the kernel sets of size K that include S:

$$C_c = \{C_M | \, s \subset C_M, \qquad |C_M| = K\}.$$

Algorithm 7.2: Simulated Annealing for Optimizing Memory Subset

1. Input:
   - Concept Lattice: L
   - Frequency Distribution: P
   - Kernel Size Constraint: K
2. Output:
   - The optimal C_M concept set
3. Algorithm:

3.1. Initialize the Kernel Set with C_A.
3.2. Set Simulated Annealing Parameters:
3.2.1. Initial Temperature T: Starting temperature (e.g., 1500.0).
3.2.2. Final Temperature T_{final}: Temperature at which the algorithm terminates (e.g., 1.0)

DOI: 10.14750/ME.2026.038

3.2.3. Cooling Rate α: Factor by which the temperature decreases each iteration (e.g., 0.95).
3.2.4. Number of Iterations per Temperature: Number of neighbor evaluations per temperature step (e.g., 200).

3.3. Generate Initial Solution:
3.3.1. Eligible Concepts $C' = C \setminus C_M$: Concepts available for selection into C_M.
3.3.2. Random Concepts Selection: Number of additional concepts to be selected.
3.3.3. Select Initial C_M: Randomly sample S_s concepts from C' and set

$$C_M = C_A \cup S_s \, .$$

3.4. Simulated Annealing Loop until $T_{initial} \leq T_{final}$:
3.4.1. Current Solution:
- Memory Subset: C'_M, the current set of selected concepts.
- Fitness: Total expected generation cost for C'_M, computed using the fitness function:

$$f_{current} = \sum\nolimits_{c \in C} p_c \, h(c).$$

3.4.2. Generate Neighbor Solution with Swap Operation:

- Remove a random concept c_o from $C_M \setminus C_A$.
- Add a random concept c_i from $C' \setminus C_M$.
- Ensure $| C_M | = K$.

3.4.3. Fitness Evaluation for Neighbor: Compute $f_{neighbor}$.
3.4.4. Calculating Acceptance Probability:
If $f_{neighbor} < f_{current}$, accept the neighbor. Otherwise, accept the neighbor with probability:

$$exp\left(-\frac{f_{\text{neighbor}} - f_{current}}{T_{initial}}\right).$$

3.4.5. Cooling Phase: Update the temperature $T = \alpha T$.
3.4.6. Termination: Stop if $T < T_{final}$.

This measure provides an estimate of the subset's relevance for building an optimal kernel set. Because directly computing $f()$ for every subset can be prohibitively expensive, we instead employ a machine-learning-based approximation strategy to predict these fitness values efficiently.

The model's output corresponds to an estimated fitness measure, serving as an approximation of $h(s)$. This estimation process employs a training set derived by uniformly sampling candidate kernel sets of size K. From these uniformly chosen samples, the algorithm compiles a training dataset that underpins the regression neural network's learning process.

Within this model, the primary objective lies in identifying suitable candidate concept subsets for both initialization and mutation stages. To achieve this, the fitness estimation is performed by a regression-oriented neural network, whose input vector v constitutes a membership representation over the concept set C. Concretely:

$v_i = 1$ if $c_i \in s$, and 0 otherwise

The model then derives a predicted fitness score, serving as an approximation to $h(s)$. To generate the training data, we begin with randomly sampled kernel sets of size K, drawing from C in a uniform manner. This process yields an initial dataset

DOI: 10.14750/ME.2026.038

$$T_0 = \{(s, h(s)) \mid s \subset C, \mid s \mid = K\},$$

From T_0, we construct a secondary training subset

$$T_1 = \{(s', h'(s')) \mid s' \subset s \in T_0\},$$

where $h'()$ denotes an aggregated fitness value computed relative to T_0. By uniting these two parts, the final training set becomes

$$T = T_0 \cup T_1.$$

The neural network utilized in our approach follows a four-layer MLP configuration, providing a sequential stack of interconnected layers as illustrated in Figure 7.1.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 200)	20200
dense_1 (Dense)	(None, 30)	6030
dense_2 (Dense)	(None, 16)	496
dense_3 (Dense)	(None, 1)	17

Total params: 26,743

Figure 7.1. MLP Framework for Fitness Approximation

During the training phase, the system recorded the changing loss value for a training dataset of 40,000 items, as depicted in Figure 7.2. The graph shows a continuous decrease in loss, indicating that the neural network effectively acquires the mapping from concept subsets to their predicted fitness values. Once the network is sufficiently trained, the algorithm proceeds to produce the most promising concept subsets by employing an apriori-based greedy procedure, grounded in the principle that strong itemsets generally contain equally strong sub-itemsets. This process begins by examining single items and estimating their predicted fitness with the trained neural network. It then advances to constructing and evaluating candidate pairs, again leveraging the neural network for selection. Following the identification of optimal pairs, the algorithm extends to forming triplets and pruning any that fail to meet performance thresholds, ultimately arriving at a refined collection of top kernel candidates as a result of this module.

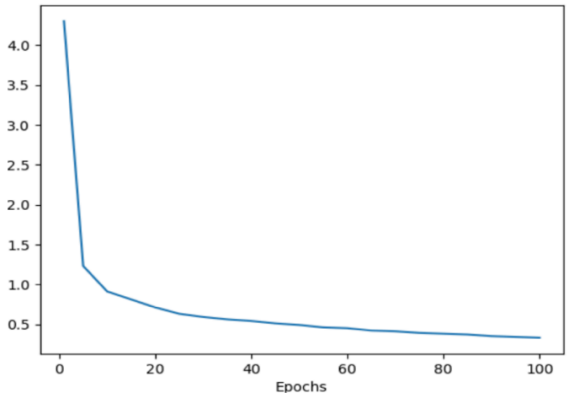


Figure 7.2. Loss Function in the Training Process

DOI: 10.14750/ME.2026.038

Subsequently, this assembled candidate set serves as the initial population for the Genetic Algorithm (GA). The impact of incorporating the relevance factor into the selection process is illustrated in Figure 7.3. In that figure, the dotted line denotes the baseline GA's performance, whereas the solid line represents the enhanced GA incorporating the relevance-based selection. Evidently, the revised algorithm persistently outperforms the baseline, underscoring the efficiency gains attributed to the relevance-based approach. Through this synergy of neural network-driven relevance prediction and a GA framework, the method accelerates the discovery of optimal kernel subsets.

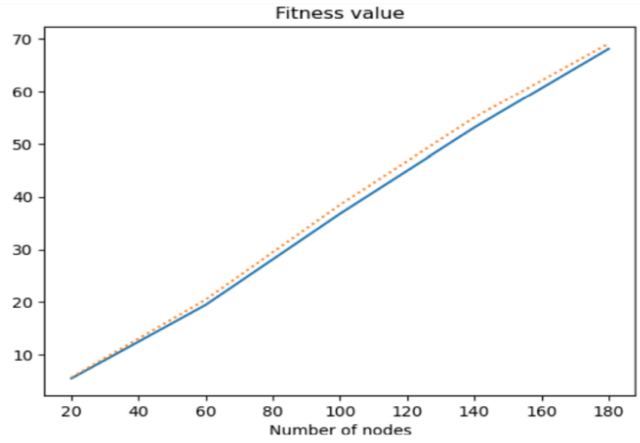


Figure 7.3. Efficiency Improvement of the Relevance-Based Selection

7.3. Practical Application in Word-Level Concept Representation

7.3.1. Problem Description

In natural language, we use words to describe the concepts that exist in our world. However, it is evident that not every concept has a dedicated single word; many concepts require more elaborate descriptions to differentiate them. In this context, words that function as "identifiers" can be thought of as memory, or kernel concepts. For other concepts, we often rely on a combination of these memory words when referring to them in conversation. Together with the kernel set, these additional concepts form the set C_F. As for any remaining concepts, we do not assign them separate expressions for unique identification. In this work, we utilize the kernel concept set mining algorithm to tackle the problem of selecting an optimal vocabulary.

To formalize this, let f be the mapping function that represents concepts at the word level:

$$f: C_F \rightarrow W^*.$$

where W^* is the set of all possible word sequences constructed from a finite collection of words W. The pool W includes the words corresponding to the kernel concepts; we denote W_c as the word linked to a specific kernel concept c.

Concerning the cost function h_{C_M}, we take a straightforward approach:

DOI: 10.14750/ME.2026.038

$$h_{C_M}(c) = |f(c)|,$$

where $|f(c)|$ indicates the length (in words) of the representation of concept c. Therefore, for every $c \in C_M$, we have

$$h_{C_M}(c) = 1.$$

If we assume C_M includes all attribute concepts $c_a = (\{a\}'', \{a\}')$ and

$$\forall a \in M: \{a\} = \{a\}'',$$

then we can specify a unique word-level representation:

$$f(c) = \{f(c_k)\} \cup \{f(c_a) | a \in attr(c) \setminus attr(c_k)\} = W_{c_k} \cup \{W_{c_a} | a \in attr(c) \setminus attr(c_k)\}.$$

where c_k denotes the nearest kernel concept to c, and $attr(c)$ is the set of attributes (the intent) of c.

Proposition 1

The above mapping function guarantees an unambiguous representation at the word level.

Proof. For any concept c, since C_M is finite and its size does not exceed K, we can identify the closest kernel concept c_k. The representation W_{c_k} is a unique word. Because $attr(c)$ and $attr(c_k)$ are individually unique, the pair $(c_k, attr(c) \setminus attr(c_k))$ yields a one-of-a-kind attribute set. Hence, the word sequence $W_{c_k} \cup \{W_{c_a} | a \in attr(c) \setminus attr(c_k)$ unambiguously denotes a specific concept in the lattice. □

Example 1

For illustration, consider the Live in Water ontology provided at: https://upriss.github.io/fca/examples.html. This ontology includes 18 concepts in total. Their frequencies are compiled in Table A.4 of Appendix A, and the frequency threshold is set at 0.4. Figure 7.4 shows the resulting concept lattice; concepts not in C_F appear with a gray background.

In this scenario, only the "specialization" operation is allowed, so

- $d(c_1, c_2) = 1$ if is a direct parent of c_2,
- $d(c_1, c_2) = \infty$ otherwise.

Using these cost settings, the kernel concept mining algorithm yields:

- Kernel concepts**:** {8, 9, 15}
- Total cost: 14.66

Within the lattice shown in Figure 7.4, these kernel concept nodes are colored orange.

DOI: 10.14750/ME.2026.038

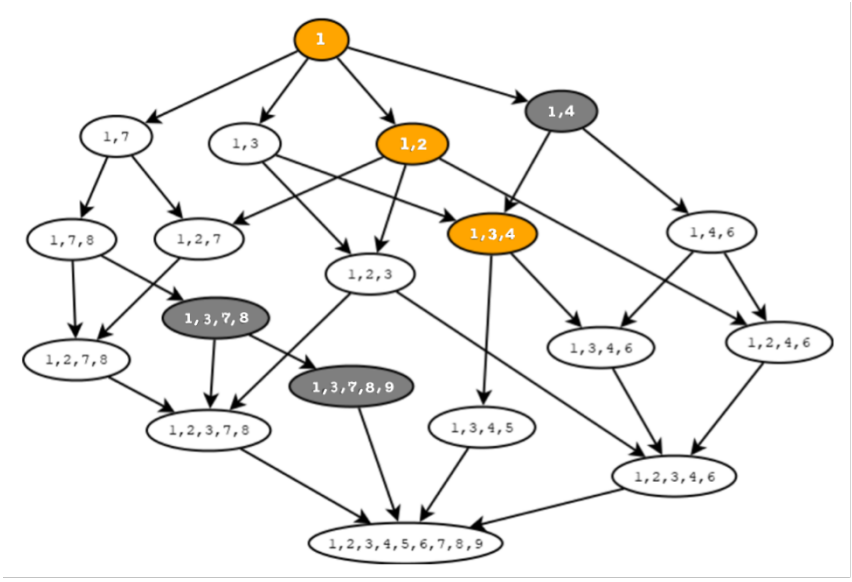


Figure 7.4. Structure of the Live in Water Ontology

7.3.2. Attribute Reduction

Although the mapping function introduced above ensures a valid word-level representation, there might be instances where some elements are redundant. In other words, certain attributes and words might be superfluous for distinguishing a particular concept, so only a subset of $attr(c) \setminus attr(c_k)$ would be needed to create an unambiguous representation. By removing these unnecessary attributes, we can streamline our overall vocabulary.

The proposed attribute reduction technique uses the attribute relevance test outlined in Algorithm 7.3. This procedure follows a greedy strategy that identifies redundant attributes in a loop. Candidate attributes are temporarily deactivated, and we check whether the remaining attributes in $attr(c) \setminus attr(c_k)$ still provide unique sets for all concepts attached to a kernel concept.

Algorithm 7.3: Attribute Reduction Algorithm

Input:
- Concept Lattice: L
- Kernel Set: P

Output:
- Reduced c_M concept set

Procedure:
1. For each kernel concept, gather all items in its cluster along with their respective sets $A(c) = attr(c) \setminus attr(c_k)$.
2. loop on all attributes a ∈ M for relevance test
   - For all concepts c and for attributes sets in $A(c)$, we remove a from the attribute sets. The result set is denoted by $A'(c)$.
   - We check, whether all sets in $A'(c)$ are unique or not.
4. . If the reduced set $A'(c)$ is unique for each concept c, then we can remove c from the kernel set

DOI: 10.14750/ME.2026.038

Example 2

Continuing the Live in Water example, we perform attribute reduction after computing the "winner" kernel concept for each concept. This computation groups concepts by kernel concept, forming separate hierarchies whose roots are the kernel concepts. Figure 7.5 visualizes these hierarchies.

Next, the algorithm pinpoints redundant attributes, and in this scenario, the attributes $\{1, 4, 9\}$ are identified as extraneous. With these removed, we obtain a reduced attribute set and reconstruct the word-level representations of all concepts. Figure 7.6 illustrates the resulting representation tree. Here, W_i denotes the word assigned to each kernel concept, while w_i stands for the words of the attribute concepts.

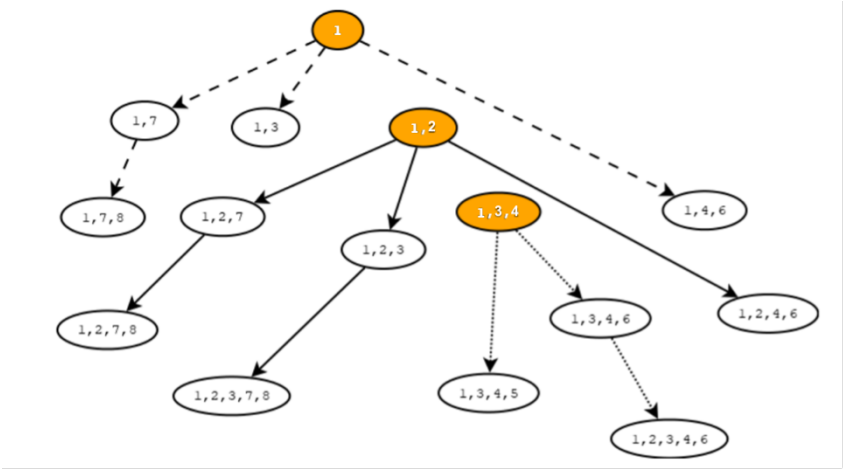


Figure 7.5. Structure of the resulted tree structures after selection of the kernel concepts

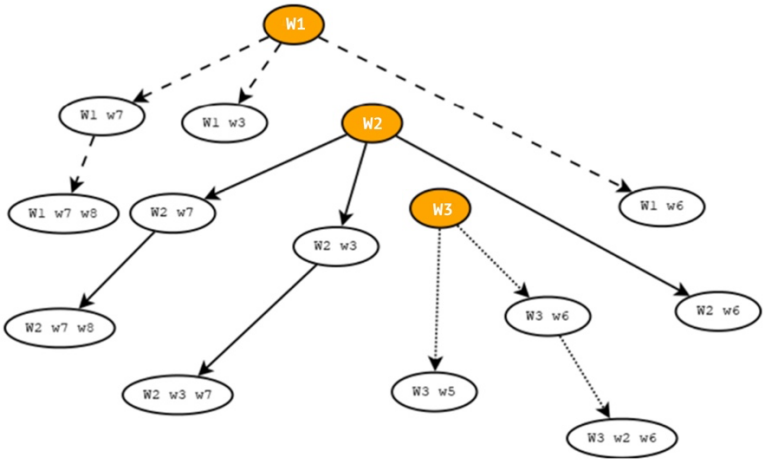


Figure 7.6. Word-Level Representation of the Concepts After Attribute Reduction

7.4. Experimental Evaluation

To validate the effectiveness of our algorithm, we implemented it in Python, selected for its extensive toolset and robust library support that streamlines the management of computationally intensive tasks. All experiments were performed on a Mac system running

DOI: 10.14750/ME.2026.038

macOS 14.3.1, equipped with an Apple M1 chip and 8 GB of RAM, thus providing a stable and resource-efficient environment for empirical assessment.

For the study's comparative analysis, we relied on four established real-world datasets obtained from the UCI Machine Learning Repository. To align these datasets with FCA, each dataset's categorical variables were translated into Boolean features, yielding formal contexts. In this process, every distinct category was mapped to a corresponding binary attribute, indicating the presence or absence of that category for a given object. Following this transformation, we constructed the respective concept lattices for each dataset, with key information summarized in Table 4.1.

The selected datasets, Balance Scale, Breast Cancer Wisconsin, Teaching Assistant Evaluation (Tae), and Car Evaluation, each exhibit unique attributes as shown in Table 4.1, such as size, attribute count, density, and structural intricacy when transformed into concept lattices. This diversity offers a thorough testbed for evaluating algorithmic performance and scalability across varied data scenarios.

This selection of datasets provides a broad and demanding environment for algorithm evaluation, enabling us to thoroughly gauge its scalability, efficiency, and overall effectiveness under varying data conditions. By incorporating sets distinguished by different sizes, numbers of attributes, and densities, we specifically challenge the algorithm's capacity to manage both large-scale and intricate lattices. High object and attribute counts probe the method's ability to handle substantial data volumes, while varying densities allow us to examine its performance in both sparse and dense configurations. Consequently, testing our model on these diverse datasets provides a robust appraisal of its effectiveness and adaptability in real-world situations that exhibit a range of complexity levels.

7.4.1. Scalability Evaluation Across Varying Lattice Dimensions

A comprehensive set of experiments was undertaken to evaluate the computational time of both the Genetic Algorithm (GA) and Simulated Annealing (SA) when applied to lattices of varying sizes. Each dataset tested features different scales in terms of object count, attribute count, and density levels. This design enables a thorough assessment of how computational time grows with increasing lattice complexity. In conducting the experiments, an exponential decay function was utilized for the probability distribution $P(s)$ over concepts, prioritizing higher-level concepts to simulate more frequent usage in natural language.

Under controlled and identical testing conditions, both GA and SA were tasked with selecting a set of core concepts for lattice reduction, consistent with the methods described in previous sections. The Genetic Algorithm and Simulated Annealing were configured as follows:

Genetic Algorithm Parameters

- Population Size (): 100
- Number of Generations (): 50
- Crossover Rate (): 0.8
- Mutation Rate (): 0.05
- Tournament Size (): 5

Simulated Annealing Parameters

DOI: 10.14750/ME.2026.038

- Initial Temperature (): 1500.0
- Final Temperature (): 1.0
- Cooling Rate (): 0.95
- Iterations per Temperature (): 200

These parameter values were chosen based on pilot studies and established practices in evolutionary algorithm research [104]. The resulting computational time for each method, as shown in Figure 7.7, indicates that the Genetic Algorithm offers substantial efficiency gains over Simulated Annealing, particularly as the concept lattice expands. While both methods see rising computational demands with larger lattices, the GA exhibits a near-linear increase in execution time. This scalability emphasizes its suitability for extensive datasets and underscores its overall advantage in handling more complex lattice structures.

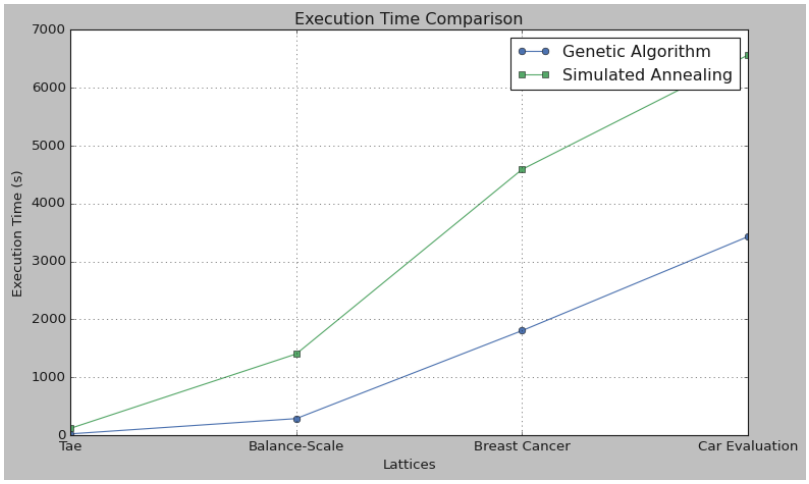


Figure 7.7. A Runtime Comparison of Genetic Algorithm (GA) and Simulated Annealing (SA) on Multiple Datasets

7.4.2. Influence of Kernel Concept Size on Overall Generation Cost

In this section, we investigate how altering the proportion of core concepts impacts both the Total Expected Generation Cost () and the overall reduction of stored concepts in a concept lattice. Two optimization algorithms, Genetic Algorithm (GA) and Simulated Annealing (SA), are evaluated at different core concept size percentages. The outcomes, presented in Table 7.1 and depicted in Figures 7.8, offer valuable insights into each algorithm's scalability and effectiveness in concept lattice reduction.

Table 7.1 Impact of Kernel Concept Size on Optimization Performance of GA and SA

Kernel Concept Size (%)	Algorithm	Core Concepts Selected	Cost of the Kernel
20.0	GA	725	2.08461
20.0	SA	725	2.09324
25.0	GA	901	1.94862
25.0	SA	901	1.96122
30.0	GA	1,077	1.83434
30.0	SA	1,077	1.84108

Our experiments specifically targeted core concept sizes of 20%, 25%, and 30% of the 3,542 formal concepts in the Car Evaluation dataset. For each chosen size, both GA and SA were tasked with identifying an optimal subset of core concepts. Their primary objective was to minimize the total generation cost () while substantially decreasing the quantity of stored concepts within the lattice.

DOI: 10.14750/ME.2026.038

To ensure rigor and consistency, the algorithmic parameters for both methods were carefully selected based on preliminary trials and recognized practices in evolutionary computation [104]. Table 7.1 demonstrates how varying the kernel concept size influences performance for both GA and SA. At a 20% kernel size, GA achieved a cost value of 2.0832, while SA recorded a marginally higher cost. Increasing the kernel size to 25% yielded respective costs of 1.9486 (GA) and 1.9612 (SA). Finally, at 30% kernel size, GA reached 1.8343, narrowly outperforming SA's 1.8418.

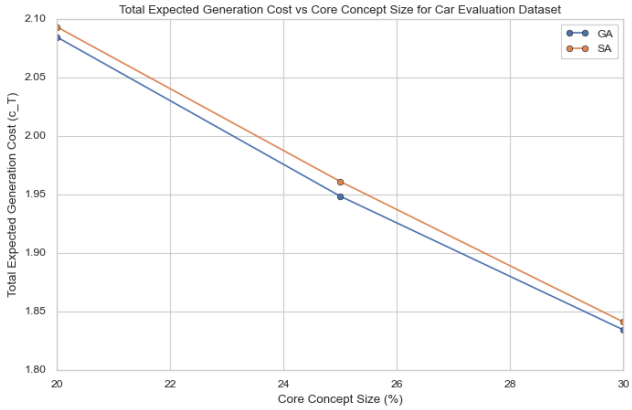


Figure 7.8. Variation of Total Generation Cost () with Kernel Concept Size (%) for GA and SA

The analysis highlights the Genetic Algorithm's (GA) strong performance in optimizing kernel concept selection, substantially improving the efficiency of concept lattice reduction. While Simulated Annealing (SA) also yields comparable cost reductions, GA's consistent advantage renders it especially suitable for scenarios where minimizing generation cost is paramount. Moreover, the marked decrease in stored concepts underscores the model's effectiveness in simplifying the lattice, making it more manageable for real-world applications. Our findings further demonstrate that enlarging the kernel concept size leads to notable decreases in the total expected generation cost, with GA consistently outperforming SA in cost-sensitive and lattice-streamlining contexts. These outcomes underscore both the scalability and robustness of the proposed model, confirming its capability to manage varying data complexities effectively in concept lattice reduction.

7.4.3. Impact of Frequency Distribution on Algorithm Performance

We conducted additional experiments by selecting kernel concept sizes of 20%, 25%, and 30% from the Tae dataset's 276 formal concepts under three distinct frequency distributions: Default, Uniform, and Random. Figures 7.9 and 7.10 illustrate that in the Default distribution, GA consistently achieved the lowest average total expected generation cost of 1.3209, with SA closely following at 1.3272. This improved outcome stems from strategically assigning high-frequency linguistic units and employing an injective mapping function, thereby streamlining the lattice while aligning with human cognitive processes by emphasizing the most frequently used concepts.

In contrast, the Uniform distribution resulted in notably higher costs, 1.7117 for GA and 1.7190 for SA, reflecting reduced optimization due to the absence of frequency-based prioritization. Meanwhile, the Random distribution yielded intermediate values of 1.6637 for GA and 1.6796 for SA, showcasing GA's resilience in adapting to stochastic frequency patterns while maintaining performance similar to the Default distribution.

DOI: 10.14750/ME.2026.038

Notably, GA also demonstrated superior runtime efficiency across all distributions, averaging about 20.65 seconds, whereas SA typically exceeded 55 seconds. Despite our model's capacity to accommodate various frequency distributions, the Default scenario proves most effective by minimizing cost while maintaining runtime efficiency. Thus, GA stands out as the preferred method in contexts demanding both cost-effectiveness and speed, particularly when exploiting structured frequency distributions that align well with natural cognitive patterns.

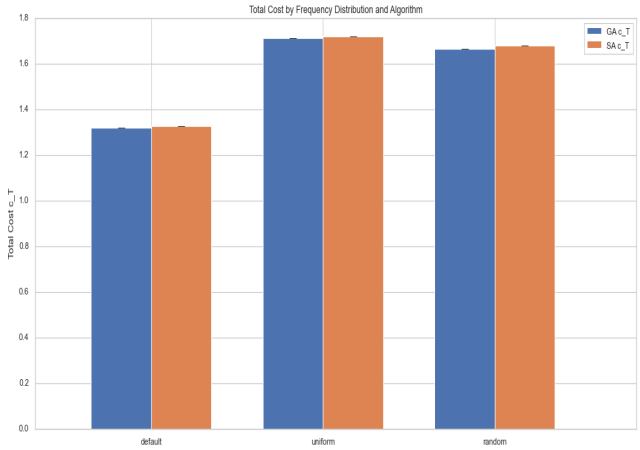


Figure 7.9. Average Cost Comparison of GA and SA Across Frequency Distributions

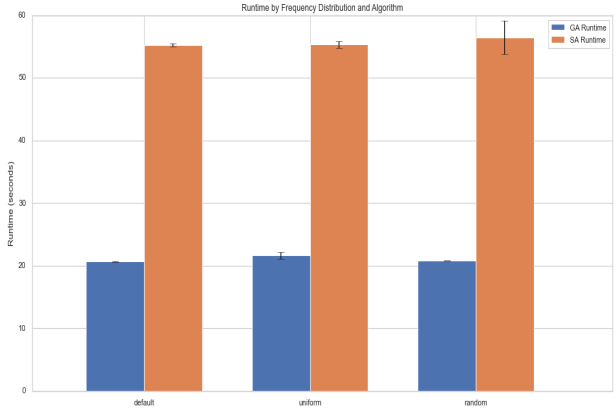


Figure 7.10. Runtime Performance of GA and SA Across Frequency Distributions

7.4.4. GA and SA Convergence in Concept Lattice Reduction

We conducted a detailed comparison of Genetic Algorithm (GA) and Simulated Annealing (SA) in reducing concept lattice complexity for a dataset containing 276 formal concepts. As depicted in Figure 7.8, each method seeks to minimize a "generation cost," which estimates the cognitive and linguistic effort needed to represent concepts. The shared objective is to streamline the lattice while preserving interpretability.

Observing Figure 7.11, SA (green line) begins at a relatively high cost and rapidly decreases, aided by its elevated initial temperature. This swift descent indicates SA's capacity to quickly identify an efficient solution, although the algorithm often stabilizes sooner, suggesting it may converge on a reasonably good, but not always optimal outcome.

DOI: 10.14750/ME.2026.038

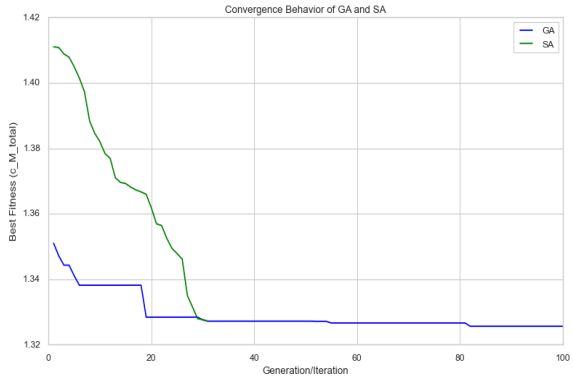


Figure 7.11. GA and SA Convergence in Concept Lattice Reduction

By contrast, GA (blue line) shows a more measured reduction in cost, attributed to its population-based framework of crossover and mutation, which continues refining solutions beyond the first stages. This extended improvement typically enables GA to arrive at a lower final cost than SA, reflecting a more thoroughly optimized solution. While both approaches effectively diminish the lattice's complexity, GA consistently achieves a slightly lower ultimate cost, aligning better with the goal of balancing expressiveness and usability in concept lattices.

Overall, the findings indicate that SA excels in rapidly yielding a near-optimal reduction, useful for scenarios demanding quick approximations, while GA's iterative refinement yields marginally superior final outcomes. Each algorithm thus caters to different priorities: SA for accelerated initial reductions and GA for achieving a more precise, cognitively aligned result.

7.5. Summary

This chapter presents an innovative approach to address the scalability and complexity hurdles in FCA. By blending cognitive insights and linguistic optimization, the proposed model strategically selects a core subset of high-frequency concepts and employs an injective mapping function. The resulting kernel subset reduces computational overhead while preserving key structural relationships in the lattice.

Comparative evaluations using Genetic Algorithms (GA) and Simulated Annealing (SA) consistently highlight GA's superior performance in both computational efficiency and minimizing generation costs, as demonstrated across multiple real-world datasets. The integration of human-centric principles not only clarifies the reduced lattice structures but also enhances usability, making FCA-based analyses more intuitive and tractable for practical applications.

Overall, the proposed methodology bridges a critical gap between efficient computational methods and cognitively aligned lattice simplifications, thereby extending the utility of FCA in complex, large-scale data environments. This intersection of cognitive efficiency and computational scalability opens new possibilities for more powerful, user-friendly lattice reduction techniques in future research and real-world implementations.

Github: https://github.com/Mdaash/KCS_Approach/blob/master/GA_and_SA_analysis.ipynb

Publications : P_3

DOI: 10.14750/ME.2026.038

Chapter 8: Conclusion

8.1. Summary

This dissertation addresses the escalating challenges of scalability and interpretability in FCA, where concept lattices can become exceedingly large as datasets grow in size and complexity. Despite FCA's robust theoretical underpinnings, traditional methods often yield unwieldy lattices that are time-consuming to compute and difficult for users to navigate. To tackle these issues, the research combines three complementary strategies for lattice reduction. First, two clustering-based algorithms, K-Means Dijkstra on Lattice (KDL) and K-Means Vector on Lattice (KVL), identify a small set of representative "centroid" concepts. KDL leverages an adapted shortest-path metric on the lattice, whereas KVL employs a vectorization step before applying k-means. Both approaches effectively compress lattice size while retaining critical structural relationships.

Building on this, the Kernel Concept Set (KCS) approach uses frequency and derivation-cost metrics to select a minimal yet structurally faithful subset of concepts. This selection-based method preserves essential patterns in the data while significantly reducing lattice complexity. Finally, a Genetic Algorithm (GA), enhanced with a neural network–based fitness evaluation, optimizes the discovery of these kernel concepts. This GA-centric strategy has been shown to outperform other benchmarks, further underscoring the robustness and efficiency of the proposed reductions.

Together, these methods provide a scalable and interpretable framework for FCA, enabling analysts to handle larger, more diverse datasets and promoting practical adoption across domains that rely on concept lattices for knowledge representation.

8.2. Contributions

The main scientific results achieved during the completion of this research are summarized below in three theses:

- Thesis 1
  Related Publications: *[P_1, P_4]*

  I have introduced two new clustering algorithms for lattice reduction in FCA: K-Means Dijkstra on Lattice (KDL) and K-Means Vector on Lattice (KVL). Both approaches adapt the standard k-means clustering framework to the specific structure of concept lattices, where the relationships between formal concepts are hierarchical rather than purely numerical.
  In the case of KDL, the method leverages a Dijkstra-based distance measure that assigns direction-sensitive costs to lattice traversal, ensuring that concept proximity is measured in terms of structural and hierarchical effort. This allows clusters to reflect the intrinsic organization of the lattice, thereby capturing semantic similarity more faithfully.
  By contrast, KVL embeds each concept into a vector space representation based on its intent and attribute frequencies. This transformation enables the direct application

DOI: 10.14750/ME.2026.038

of standard k-means clustering, providing a faster and more computationally scalable alternative while still preserving meaningful groupings.
Experimental evaluations conducted on benchmark datasets from the UCI Machine Learning Repository demonstrated that both KDL and KVL improve the balance between fidelity of conceptual structure and scalability of computation. KDL was shown to be particularly effective in producing structure-aware clusters, while KVL provided a robust and efficient method for handling larger datasets. Together, these two algorithms extend FCA into the realm of modern clustering applications, offering practical solutions for concept lattice reduction.

- Thesis 2
  Related Publications: *[P_2, P_1, P_4]*

  I have introduced the Kernel Concept Set (KCS) approach, a selection-based strategy for reducing concept lattices by identifying a small but representative subset of formal concepts. This approach is original to the present research and defines kernel concepts as those that combine high frequency of occurrence with low derivation cost, making them both semantically central and computationally efficient.
  KCS thus balances two competing objectives: preserving interpretability while reducing computational complexity. By retaining kernel concepts as structural "anchors," the lattice can be effectively approximated without losing essential relationships. This represents a departure from earlier methods such as iceberg lattices, which rely solely on frequency thresholds and therefore risk discarding structurally important but less frequent concepts.
  Comparative experiments confirmed that KCS yields smaller, more interpretable lattices while still covering the most significant conceptual structures. Furthermore, the approach enhances usability by aligning with human cognitive processes of focusing on "core" concepts, making the reduced lattices easier to visualize and analyze. In this way, KCS offers both theoretical novelty and practical utility, bridging a gap between efficiency and semantic clarity in lattice reduction.

- Thesis 3
  Related Publications: *[P_3]*

  I proposed an optimized Genetic Algorithm (GA) solution for mining kernel concepts in FCA. This method introduces a hybrid strategy where the GA is enhanced by a neural network module to accelerate fitness evaluation, thereby reducing the computational overhead typically associated with evolutionary approaches. The genetic optimization process was specifically tailored to select kernel sets that minimize overall derivation cost while respecting constraints on set size and interpretability. Through extensive testing on benchmark datasets, the GA-based method consistently outperformed existing approaches in terms of both efficiency and quality of selected kernel sets.
  Beyond pure efficiency, the method also demonstrated adaptability to application domains such as computational linguistics, where kernel concepts can be used to

DOI: 10.14750/ME.2026.038

represent core semantic structures in textual data. This illustrates the broader potential of kernel-based reduction beyond formal lattice theory, highlighting its utility in interdisciplinary research contexts.

Taken together, these three theses establish a coherent research program that advances the state of the art in Formal Concept Analysis. By introducing two novel clustering methods (KDL and KVL), formulating the original concept of Kernel Concept Sets, and designing an optimized evolutionary algorithm for kernel selection, this dissertation provides a comprehensive framework for scalable and interpretable lattice reduction. The results open pathways for applying FCA to increasingly complex and large-scale data, bridging theory, computation, and real-world application.

8.3. Future Works

The methods introduced in this dissertation kernel concepts, the Dijkstra-based distance measure, and the clustering frameworks KDL and KVL provide an initial but promising foundation for reducing the size and complexity of concept lattices while maintaining interpretability. Nevertheless, several avenues exist for extending and improving these contributions in future work. One important direction lies in advancing the kernel concept framework. While the current approach balances frequency and derivation cost, future research may design richer cost functions that integrate semantic weights, probabilistic relevance, or user-defined priorities. In addition, adaptive selection strategies could be developed, where the kernel set dynamically adjusts according to the analytical task (e.g., association rule mining vs. clustering), thus making kernel concepts even more versatile as structural anchors in FCA.

The Dijkstra-based distance measure also opens room for refinement. At present, fixed costs are assigned to upward and downward lattice moves. Future studies could investigate adaptive or data-driven weighting schemes, where the costs are learned from data distributions or domain-specific feedback. Moreover, exploring approximations of shortest paths such as pruning uninformative regions of the lattice or using heuristic accelerators may enhance scalability without sacrificing structural fidelity.

For the KDL clustering method, one limitation arises from the repeated execution of shortest-path computations. Future work could incorporate advanced indexing strategies, parallel graph processing, or precomputed distance matrices to mitigate this overhead. Beyond efficiency, the algorithm itself may benefit from hybridization with other clustering paradigms, such as density-based methods or spectral clustering, allowing KDL to capture different structural properties of the lattice. Similarly, the KVL method could be expanded by refining its vectorization strategy. Currently, attribute frequencies are used to approximate absent attributes; however, incorporating global statistical measures (such as mutual information) or embedding-based representations could yield vectors that capture more subtle semantic similarities. This could improve clustering accuracy while maintaining computational efficiency.

Beyond individual methods, future work should also explore the integration of kernel concepts and lattice-based distances with machine learning pipelines. For example, kernel sets could serve as interpretable features in classification tasks, or lattice-based distances could enhance similarity measures in recommender systems. Embedding these methods into hybrid symbolic–statistical frameworks would not only extend their applicability but also strengthen the interpretability of AI systems, a concern of growing importance in

DOI: 10.14750/ME.2026.038

contemporary research. Finally, a promising direction lies in extending experimental validation. While the current evaluation relied on UCI benchmark datasets, applying the methods to larger, real-world contexts (e.g., biomedical ontologies, legal knowledge graphs, or e-commerce transaction data) would both test scalability and demonstrate practical utility. Such applications could highlight the unique contribution of kernel concepts and lattice-aware clustering in domains where transparency, efficiency, and semantic structure are equally critical.

DOI: 10.14750/ME.2026.038

Author's Publications

Publications Related to the Dissertation

[P_1] M. Alwersh and L. Kovács, "K-Means Extensions for Clustering Categorical Data on Concept Lattice," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 9, 2023. ***Scopus Indexed [Q3]***.

[P_2] ALWERSH, Mohammed; KOVÁCS, László. Enhancing Formal Concept Analysis with the Kernel Concept Set Approach: A Novel Methodology for Efficient Lattice Reduction. International Journal of Intelligent Engineering & Systems, 2024, 17.4. ***Scopus Indexed [Q2]***.

[P_3] L. Kovács and M. Alwersh, "Mining of Kernel Concepts based on Optimization of Concept Set Generation Costs," *Knowledge and Information Systems*, **manuscript submitted for publication and currently under peer review**. ***Scopus Indexed [Q1]***.

[P_4] M. Alwersh and L. Kovács, "Survey on attribute and concept reduction methods in formal concept analysis," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 1, pp. 366–387, Apr. 2023, doi: 10.11591/ijeecs.v30.i1.pp366-387. . ***Scopus Indexed [Q3]***.

Other Publications Journal Articles in non-Q Ranking

[P_5] ALWERSH, Mohammed; KOVÁCS, László. Fuzzy formal concept analysis: approaches, applications and issues. *Computer Science and Information Technologies*, 2022, 3.2: 126-136.

[P_6] ALWERSH, Mohammed. Integration of FCA with Fuzzy logic: a survey. *Multidiszciplináris Tudományok*, 2021, 11.5: 373-385.

DOI: 10.14750/ME.2026.038

References

[1] R. Wille, “Restructuring lattices theory: an approach on hierarchies of concepts,” 1982, Dordrecht, Holland: Springer.

[2] S. Roscoe, M. Khatri, A. Voshall, S. Batra, S. Kaur, and J. Deogun, “Formal concept analysis applications in bioinformatics,” ACM Comput Surv, vol. 55, no. 8, pp. 1–40, 2022.

[3] G. G. Aadil and D. Samad, “Exploring the Impact of Informal Language on Sentiment Analysis Models for Social Media Text Using Convolutional Neural Networks,” Multidiszciplináris Tudományok, vol. 13, no. 1, pp. 244–254, 2023.

[4] G. H. A. Ahmed, J. Alshboul, and L. Kovács, “Development of Ontology-based Domain Knowledge Model for IT Domain in e-Tutor Systems,” International Journal of Advanced Computer Science and Applications, vol. 13, no. 5, 2022.

[5] H. A. A. Ghanim and L. Kovács, “Ontology Supported Domain Knowledge Module for E-Tutoring System,” Acta Cybernetica, vol. 26, no. 3, pp. 455–474, 2024.

[6] K. Sumangali and C. A. Kumar, “Critical analysis on open source LMSs using FCA,” International Journal of Distance Education Technologies (IJDET), vol. 11, no. 4, pp. 97–111, 2013, doi: 10.4018/ijdet.2013100107.

[7] Y. Liu and X. Li, “Application of formal concept analysis in association rule mining,” in 2017 4th International Conference on Information Science and Control Engineering (ICISCE), IEEE, 2017, pp. 203–207.

[8] K. Sumangali and C. Aswani Kumar, “Knowledge reduction in formal contexts through CUR matrix decomposition,” Cybern Syst, vol. 50, no. 5, pp. 465–496, 2019, doi: 10.1080/01969722.2019.1602300.

[9] S. Hao, C. Shi, Z. Niu, and L. Cao, “Concept coupling learning for improving concept lattice-based document retrieval,” Eng Appl Artif Intell, vol. 69, pp. 65–75, 2018, doi: 10.1016/j.engappai.2017.12.007.

[10] R. Ganter and R. Wille, “Formal concept analysis: Mathematical foundations Springer-Verlag Berlin Germany,” 1999.

[11] B. A. Davey, Introduction to Lattices and Order. Cambridge University Press, 2002.

[12] F. Hao, Y. Yang, G. Min, and V. Loia, “Incremental construction of three-way concept lattice for knowledge discovery in social networks,” Inf Sci (N Y), vol. 578, pp. 257–280, 2021, doi: 10.1016/j.ins.2021.07.031.

[13] F. Hao, Y. Yang, B. Pang, N. Y. Yen, and D.-S. Park, “A fast algorithm on generating concept lattice for symmetry formal context constructed from social networks,” J Ambient Intell Humaniz Comput, pp. 1–8, 2019, doi: 10.1007/s12652-019-01274-6.

[14] Y. Yang, F. Hao, B. Pang, G. Min, and Y. Wu, “Dynamic maximal cliques detection and evolution management in social internet of things: A formal concept analysis approach,” IEEE Trans Netw Sci Eng, vol. 9, no. 3, pp. 1020–1032, 2021, doi: doi:10.1109/TNSE.2021.3067939.

[15] J. Poelmans, D. I. Ignatov, S. O. Kuznetsov, and G. Dedene, “Formal concept analysis in knowledge processing: A survey on applications,” Expert Syst Appl, vol. 40, no. 16, pp. 6538–6560, 2013.

[16] J. Baixeries, L. Szathmary, P. Valtchev, and R. Godin, “Yet a faster algorithm for building the Hasse diagram of a concept lattice,” in Formal Concept Analysis: 7th International Conference, ICFCA 2009 Darmstadt, Germany, May 21-24, 2009 Proceedings 7, Springer, 2009, pp. 162–177. doi: https://doi.org/10.1007/978-3-642-01815-2_13.

[17] K. Sumangali and C. A. Kumar, “A comprehensive overview on the foundations of formal concept analysis,” Knowledge Management & E-Learning: An International Journal, vol. 9, no. 4, pp. 512–538, 2017.

[18] M. Alwersh and L. Kovács, “Survey on attribute and concept reduction methods in formal concept analysis,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 30, no. 1, pp. 366–387, Apr. 2023, doi: 10.11591/ijeecs.v30.i1.pp366-387.

[19] C. Carpineto, Concept Data Analysis: Theory and Applications. Wiley, 2004.

[20] S. O. Kuznetsov and S. A. Obiedkov, “Comparing performance of algorithms for generating concept lattices,” Journal of Experimental & Theoretical Artificial Intelligence, vol. 14, no. 2–3, pp. 189–216, 2002.

DOI: 10.14750/ME.2026.038

[21] Bernhard Ganter, “Two basic algorithms in concept analysis. FB4- Preprint No 831, 1984.”.
[22] S. O. Kuznetsov and S. A. Obiedkov, “Comparing performance of algorithms for generating concept lattices,” Journal of Experimental & Theoretical Artificial Intelligence, vol. 14, no. 2–3, pp. 189–216, 2002.
[23] J. Outrata and V. Vychodil, “Fast algorithm for computing fixpoints of Galois connections induced by object-attribute relational data,” Inf Sci (N Y), vol. 185, no. 1, pp. 114–127, 2012.
[24] S. Andrews, “In-close, a fast algorithm for computing formal concepts,” in Proceedings of the 17th International Conference on Conceptual Structures (ICCS 2009), Moscow, Russia, Jul. 2009.
[25] S. Andrews, “In-close2, a high performance formal concept miner,” in International Conference on Conceptual Structures, Springer, 2011, pp. 50–62.
[26] E. M. Norris, “An algorithm for computing the maximal rectangles in a binary relation,” Revue Roumaine de Mathématiques Pures et Appliquées, vol. 23, no. 2, pp. 243–250, 1978.
[27] D. Van Der Merwe, S. Obiedkov, and D. Kourie, “Addintent: A new incremental algorithm for constructing concept lattices,” in Concept Lattices: Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004. Proceedings 2, Springer, 2004, pp. 372–385.
[28] L. Kovács, “Efficiency analsyis of concept lattice construction algorithms,” Procedia Manuf, vol. 22, pp. 11–18, 2018, doi: https://doi.org/10.1016/j.promfg.2018.03.003.
[29] P. Valtchev, R. Missaoui, R. Godin, and M. Meridji, “Generating frequent itemsets incrementally: two novel approaches based on Galois lattice theory,” Journal of Experimental & Theoretical Artificial Intelligence, vol. 14, no. 2–3, pp. 115–142, 2002.
[30] P. Valtchev, M. R. Hacene, and R. Missaoui, “A generic scheme for the design of efficient on-line algorithms for lattices,” in International conference on conceptual structures, Springer, 2003, pp. 282–295.
[31] P. Valtchev, R. Missaoui, and R. Godin, “Formal concept analysis for knowledge discovery and data mining: The new challenges,” in International conference on formal concept analysis, Springer, 2004, pp. 352–371.
[32] P. Valtchev, R. Missaoui, and P. Lebrun, “A partition-based approach towards constructing Galois (concept) lattices,” Discrete Math, vol. 256, no. 3, pp. 801–829, 2002.
[33] P. Valtchev and R. Missaoui, “Building concept (Galois) lattices from parts: generalizing the incremental methods,” in International Conference on Conceptual Structures, Springer, 2001, pp. 290–303.
[34] R. Godin, R. Missaoui, and H. Alaoui, “Incremental concept formation algorithms based on Galois (concept) lattices,” Comput Intell, vol. 11, no. 2, pp. 246–267, 1995.
[35] A. Körei and S. Radeleczki, “Box elements in a concept lattice,” Contributions to ICFCA, vol. 2006, pp. 41–56, 2006.
[36] S. Radeleczki and L. Veres, “An incremental method for the construction of the box extents of a context,” Math. Appl, vol. 10, pp. 71–78, 2021.
[37] D. Gégény, L. Kovács, and S. Radeleczki, “Notes on the lattice of fuzzy rough sets with crisp reference sets,” International Journal of Approximate Reasoning, vol. 126, pp. 124–132, 2020.
[38] A. Körei, “Using Formal Concept Analysis in the Evaluation Process,” in Teaching and Learning in a Digital World: Proceedings of the 20th International Conference on Interactive Collaborative Learning–Volume 2, Springer, 2018, pp. 143–149.
[39] A. Körei, “Applying formal concept analysis in machine-part grouping problems,” in 2013 IEEE 11th International Symposium on Applied Machine Intelligence and Informatics (SAMI), IEEE, 2013, pp. 197–200.
[40] L. Kovács and S. Gábor, “Generalization of String Transformation Rules using Optimized Concept Lattice Construction Method,” Procedia Eng, vol. 181, pp. 604–611, 2017.
[41] U. Priss, “Lattice-based information retrieval,” KO Knowledge Organization, vol. 27, no. 3, pp. 132–142, 2000.
[42] C. Carpineto and G. Romano, “Using concept lattices for text retrieval and mining,” in Formal Concept Analysis: foundations and applications, Springer, 2005, pp. 161–179.
[43] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal, “Computing iceberg concept lattices with titanic,” Data Knowl Eng, vol. 42, no. 2, pp. 189–222, 2002.

DOI: 10.14750/ME.2026.038

[44] S. Andrews and C. Orphanides, “Analysis of large data sets using formal concept lattices,” in , Proceedings of the 7th International Conference on Concept Lattices and Their Applications (CLA 2010, University of Seville, Seville, Spain, 2010, pp. 104–115.
[45] C. A. Kumar and S. Srinivas, “Concept lattice reduction using fuzzy K-means clustering,” Expert Syst Appl, vol. 37, no. 3, pp. 2696–2704, 2010.
[46] M. Alwersh and L. Kovács, “K-Means Extensions for Clustering Categorical Data on Concept Lattice,” International Journal of Advanced Computer Science and Applications, vol. 14, no. 9, 2023.
[47] L. Kovács, “Conceptual clustering with application on FCA context,” Expert Syst Appl, vol. 245, p. 123013, 2024.
[48] V. Snásel, M. Polovincak, H. M. D. Abdulla, and Z. Horak, “On Concept Lattices and Implication Bases from Reduced Contexts.,” ICCS supplement, vol. 8, pp. 83–90, 2008.
[49] N. Messai, M.-D. Devignes, A. Napoli, and M. Smail-Tabbone, “Many-valued concept lattices for conceptual clustering and information retrieval,” in ECAI 2008, IOS Press, 2008, pp. 127–131.
[50] R. G. Pensa and J.-F. Boulicaut, “Towards fault-tolerant formal concept analysis,” in Congress of the Italian Association for Artificial Intelligence, Springer, 2005, pp. 212–223.
[51] P. Krajca and V. Vychodil, “Distributed algorithm for computing formal concepts using map-reduce framework,” in International Symposium on Intelligent Data Analysis, Springer, 2009, pp. 333–344.
[52] A. Gupta, V. Bhatnagar, and N. Kumar, “Mining closed itemsets in data stream using formal concept analysis,” in Data Warehousing and Knowledge Discovery: 12th International Conference, DAWAK 2010, Bilbao, Spain, August/September 2010. Proceedings 12, Springer, 2010, pp. 285–296.
[53] S. A. Yevtushenko, “System of data analysis" Concept Explorer,” in Proc. 7th National Conference on Artificial Intelligence (KIF00), 2000, pp. 127–134. Accessed: Dec. 21, 2024. [Online]. Available: version 1.3. [Online]. Available: http://sourceforge.net/projects/conexp
[54] P. Valtchev, D. Grosser, C. Roume, and M. R. Hacene, “Galicia: an open platform for lattices,” in Using Conceptual Structures: Contributions to 11th Intl. Conference on Conceptual Structures, 2003, pp. 241–254.
[55] M. Priya and A. K. Ch, “A novel method for merging academic social network ontologies using formal concept analysis and hybrid semantic similarity measure,” Library Hi Tech, 2019, doi: 10.1108/LHT-02-2019-0035.
[56] S. M. Dias and N. J. Vieira, “Concept lattices reduction: Definition, analysis and classification,” Expert Syst Appl, vol. 42, no. 20, pp. 7084–7097, 2015.
[57] J. Medina, “Relating attribute reduction in formal, object-oriented and property-oriented concept lattices,” Computers & Mathematics with Applications, vol. 64, no. 6, pp. 1992–2002, 2012, doi: 10.1016/j.camwa.2012.03.087.
[58] J. Li, C. Mei, and Y. Lv, “A heuristic knowledge-reduction method for decision formal contexts,” Computers & Mathematics with Applications, vol. 61, no. 4, pp. 1096–1106, 2011, doi: 10.1016/j.camwa.2010.12.060.
[59] S. Peng and A. Yamamoto, “Concept Lattice Reduction Using Integer Programming,” Knowledge-based systems research group/Japan Society for Artificial Intelligence [edited], vol. 123, pp. 38–43, 2021.
[60] W. X. Zhang, L. Wei, and J. J. Qi, “Reduction Theory and Approach to Concept Lattice. China Ser,” E Inform. Sci, vol. 35, pp. 628–639, 2005, doi: 10.1360/122004-104.
[61] J.-J. Qi, “Attribute reduction in formal contexts based on a new discernibility matrix,” J Appl Math Comput, vol. 30, no. 1, pp. 305–314, 2009, doi: 10.1007/s12190-008-0174-9.
[62] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a review,” ACM computing surveys (CSUR), vol. 31, no. 3, pp. 264–323, 1999.
[63] K. S. K. Cheung and D. Vogel, “Complexity reduction in lattice-based information retrieval,” Inf Retr Boston, vol. 8, pp. 285–299, 2005.
[64] H. Yang, K. Qin, Q. Hu, and L. Yang, “Neighborhood based concept lattice,” Applied Intelligence, vol. 53, no. 5, pp. 6025–6040, 2023.
[65] C. Wang, Y. Bo, and C. Xu, “Attribute reduction algorithm on concept lattice and application in smart city energy consumption analysis,” Wirel Commun Mob Comput, vol. 2022, 2022.
[66] S. Zhao, J. Qi, J. Li, and L. Wei, “Concept reduction in formal concept analysis based on representative concept matrix,” International Journal of Machine Learning and Cybernetics, vol. 14, no. 4, pp. 1147–1160, 2023.

[67] K. Pang, P. Liu, S. Li, L. Zou, M. Lu, and L. Martínez, "Concept lattice simplification with fuzzy linguistic information based on three-way clustering," International Journal of Approximate Reasoning, vol. 154, pp. 149–175, 2023.
[68] M. Akram, H. S. Nawaz, and M. Deveci, "Attribute reduction and information granulation in Pythagorean fuzzy formal contexts," Expert Syst Appl, vol. 222, p. 119794, 2023.
[69] R. Belohlavek and V. Vychodil, "Formal concept analysis with background knowledge: attribute priorities," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 39, no. 4, pp. 399–409, 2009.
[70] R. Bělohlávek, V. Sklenář, and J. Zacpal, "Formal concept analysis with hierarchically ordered attributes," Int J Gen Syst, vol. 33, no. 4, pp. 383–394, 2004, doi: 10.1080/03081070410001679715.
[71] Z. Wang, C. Shi, L. Wei, and Y. Yao, "Tri-granularity attribute reduction of three-way concept lattices," Knowl Based Syst, vol. 276, p. 110762, 2023.
[72] R. Belohlávek, V. Sklenar, and J. Zacpal, "Concept Lattices Constrained by Attribute Dependencies.," in DATESO, Citeseer, 2004, pp. 63–73.
[73] J.-F. Boulicaut and J. Besson, "Actionability and formal concepts: A data mining perspective," in International Conference on Formal Concept Analysis, Springer, 2008, pp. 14–31.
[74] L. PISKOVÁ, T. HORVÁTH, and S. KRAJČI, "RANKING FORMAL CONCEPTS BY UTILIZING MATRIX FACTORIZATION.," Studia Universitatis Babes-Bolyai, Informatica, vol. 59, 2014.
[75] V. Ganti, J. Gehrke, and R. Ramakrishnan, "CACTUS—clustering categorical data using summaries," in Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 1999, pp. 73–83.
[76] S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," Inf Syst, vol. 25, no. 5, pp. 345–366, Jul. 2000, doi: 10.1016/S0306-4379(00)00022-3.
[77] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," Data Min Knowl Discov, vol. 2, no. 3, pp. 283–304, 1998.
[78] Z. Huang, "Clustering large data sets with mixed numeric and categorical values," in Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(PAKDD), Citeseer, 1997, pp. 21–34.
[79] D. Ienco, R. G. Pensa, and R. Meo, "Context-based distance learning for categorical data clustering," in Advances in Intelligent Data Analysis VIII: 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31-September 2, 2009. Proceedings 8, Springer, 2009, pp. 83–94. doi: https://doi.org/10.1007/978-3-642-03915-7_8.
[80] J. B. McQueen, "Some methods of classification and analysis of multivariate observations," in Proc. of 5th Berkeley Symposium on Math. Stat. and Prob., 1967, pp. 281–297.
[81] O. M. San, V.-N. Huynh, and Y. Nakamori, "An alternative extension of the k-means algorithm for clustering categorical data," International journal of applied mathematics and computer science, vol. 14, no. 2, pp. 241–247, 2004.
[82] L. Chen and S. Wang, "Central clustering of categorical data with automated feature weighting," in Twenty-Third International Joint Conference on Artificial Intelligence, Citeseer, 2013.
[83] Z. Huang and M. K. Ng, "A fuzzy k-modes algorithm for clustering categorical data," IEEE transactions on Fuzzy Systems, vol. 7, no. 4, pp. 446–452, 1999, doi: 10.1109/91.784206.
[84] F. Cao, J. Liang, D. Li, L. Bai, and C. Dang, "A dissimilarity measure for the k-Modes clustering algorithm," Knowl Based Syst, vol. 26, pp. 120–127, 2012, doi: https://doi.org/10.1016/j.knosys.2011.07.011.
[85] M. Li, S. Deng, L. Wang, S. Feng, and J. Fan, "Hierarchical clustering algorithm for categorical data using a probabilistic rough set model," Knowl Based Syst, vol. 65, pp. 60–71, 2014, doi: https://doi.org/10.1016/j.knosys.2014.04.008.
[86] G. K. Zipf, The Principle of Least Effort. CH3, 1949.
[87] S. T. Piantadosi, H. Tily, and E. Gibson, "Word lengths are optimized for efficient communication," Proceedings of the National Academy of Sciences, vol. 108, no. 9, pp. 3526–3529, 2011.
[88] R. F. I. Cancho and R. V Solé, "Least effort and the origins of scaling in human language," Proceedings of the National Academy of Sciences, vol. 100, no. 3, pp. 788–791, 2003.
[89] T. Abiy, H. Pang, C. Williams, J. Khim, and E. Ross, "Dijkstra's shortest path algorithm," Retrieved from, 2016.

DOI: 10.14750/ME.2026.038

[90] F. Mukhlif and A. Saif, “Comparative study on Bellman-Ford and Dijkstra algorithms,” in Int. Conf. Comm. Electric Comp. Net, 2020.
[91] R. Bellman, “On a routing problem,” Q Appl Math, vol. 16, no. 1, pp. 87–90, 1958.
[92] R. W. Floyd, “Algorithm 97: shortest path,” Commun ACM, vol. 5, no. 6, p. 345, 1962.
[93] D. B. Johnson, “Efficient algorithms for shortest paths in sparse networks,” Journal of the ACM (JACM), vol. 24, no. 1, pp. 1–13, 1977.
[94] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” IEEE transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.
[95] R. Poli, M. Healy, and A. Kameas, Theory and applications of ontology: Computer applications. Springer, 2010.
[96] I. Horrocks, “Owl: A description logic based ontology language,” in International conference on principles and practice of constraint programming, Springer, 2005, pp. 5–8.
[97] A. Pérez-Suárez, J. F. Martínez-Trinidad, and J. A. Carrasco-Ochoa, “A review of conceptual clustering algorithms,” Artif Intell Rev, vol. 52, pp. 1267–1296, 2019.
[98] D. Fisher, “A hierarchical conceptual clustering algorithm,” 1985.
[99] F. Beil, M. Ester, and X. Xu, “Frequent term-based text clustering,” in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002, pp. 436–442.
[100] R. C. Romero-Zaliz, C. Rubio-Escudero, J. P. Cobb, F. Herrera, O. Cordón, and I. Zwir, “A multiobjective evolutionary conceptual clustering methodology for gene annotation within structural databases: a case of study on the gene ontology database,” IEEE Transactions on Evolutionary Computation, vol. 12, no. 6, pp. 679–701, 2008.
[101] D. H. Fisher, “Knowledge acquisition via incremental conceptual clustering,” Mach Learn, vol. 2, pp. 139–172, 1987.
[102] T.-H. T. Nguyen and V.-N. Huynh, “A k-means-like algorithm for clustering categorical data using an information theoretic-based dissimilarity measure,” in Foundations of Information and Knowledge Systems: 9th International Symposium, FoIKS 2016, Linz, Austria, March 7-11, 2016. Proceedings 9, Springer, 2016, pp. 115–130. doi: https://doi.org/10.1007/978-3-319-30024-5_7.
[103] D. Schütt, “Abschätzungen für die Anzahl der Begriffe von Kontexten,” Master’s Thesis, TH Darmstadt, 1987.
[104] R. L. Haupt, “Practical genetic algorithms,” 2004, John Wiley & Sons, Inc.

DOI: 10.14750/ME.2026.038

Appendix

A. 1. The Formal Concepts Derived from the Cross-Table Described in Table 2.3

Concept #	Extent (X)	Intent (Y)	Formal Concept (X,Y)
1	$\emptyset$	{T, B, S, D, L, M, G}	($\emptyset$, {T, B, S, D, L, M, G})
2	$\{L_1\}$	{T, B, S, L}	($\{L_1\}$, {T, B, S, L})
3	$\{L_2\}$	{B, S, M}	($\{L_2\}$, {B, S, M})
4	$\{L_3\}$	{T, S, D, G}	($\{L_3\}$, {T, S, D, G})
5	$\{L_5\}$	{T, B, L, M}	($\{L_5\}$, {T, B, L, M})
6	$\{L_7\}$	{S, L, M, G}	($\{L_7\}$, {S, L, M, G})
7	$\{L_8\}$	{T, B, D, L, G}	($\{L_8\}$, {T, B, D, L, G})
8	$\{L_1, L_2\}$	{B, S}	($\{L_1, L_2\}$, {B, S})
9	$\{L_1, L_3\}$	{T, S}	($\{L_1, L_3\}$, {T, S})
10	$\{L_1, L_7\}$	{S, L}	($\{L_1, L_7\}$, {S, L})
11	$\{L_2, L_5\}$	{B, M}	($\{L_2, L_5\}$, {B, M})
12	$\{L_2, L_7\}$	{S, M}	($\{L_2, L_7\}$, {S, M})
13	$\{L_3, L_7\}$	{S, G}	($\{L_3, L_7\}$, {S, G})
14	$\{L_3, L_8\}$	{T, D, G}	($\{L_3, L_8\}$, {T, D, G})
15	$\{L_5, L_7\}$	{L, M}	($\{L_5, L_7\}$, {L, M})
16	$\{L_6, L_8\}$	{T, D, L}	($\{L_6, L_8\}$, {T, D, L})
17	$\{L_7, L_8\}$	{L, G}	($\{L_7, L_8\}$, {L, G})
18	$\{L_1, L_5, L_8\}$	{T, B, L}	($\{L_1, L_5, L_8\}$, {T, B, L})
19	$\{L_2, L_5, L_7\}$	{M}	($\{L_2, L_5, L_7\}$, {M})
20	$\{L_3, L_6, L_8\}$	{T, D}	($\{L_3, L_6, L_8\}$, {T, D})
21	$\{L_3, L_7, L_8\}$	{G}	($\{L_3, L_7, L_8\}$, {G})
22	$\{L_1, L_2, L_5, L_8\}$	{B}	($\{L_1, L_2, L_5, L_8\}$, {B})
23	$\{L_1, L_5, L_6, L_8\}$	{T, L}	($\{L_1, L_5, L_6, L_8\}$, {T, L})
24	$\{L_1, L_2, L_3, L_4, L_7\}$	{S}	($\{L_1, L_2, L_3, L_4, L_7\}$, {S})
25	$\{L_1, L_3, L_5, L_6, L_8\}$	{T}	($\{L_1, L_3, L_5, L_6, L_8\}$, {T})
26	$\{L_1, L_5, L_6, L_7, L_8\}$	{L}	($\{L_1, L_5, L_6, L_7, L_8\}$, {L})

DOI: 10.14750/ME.2026.038

27	$\{L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8\}$	$\emptyset$	$(\{L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8\}, \emptyset)$

A. 2. Kernel Concept Set Analysis of TA Assignments (S_c set to 5%)

Concept ID	Number of TAs Sharing Attributes	Highlighted Attributes
1	2	Course_3, Summer, Course_Instructor_15, Class_Size_17, Eng_Nat_spk_2
2	2	Class_Size_19, Course_3, Summer, Course_Instructor_23, Eng_Nat_spk_1
3	3	regular, Eng_Nat_spk_2, Course_1, Class_Size_51
4	3	Summer_or_regular_2, Eng_Nat_spk_2, Course_3, Course_Instructor_8
5	3	Summer_or_regular_2, Eng_Nat_spk_2, Course_5, Course_Instructor_9
6	3	Summer_or_regular_2, Course_3, Course_Instructor_22, Eng_Nat_spk_1
7	4	Course_7, Eng_Nat_spk_2, Summer_or_regular_2, Course_Instructor_25
8	4	Summer_or_regular_2, Eng_Nat_spk_2, Course_3, Course_Instructor_23
9	6	Eng_Nat_spk_2, Class_Size_20, Summer_or_regular_1, Course_3
10	7	Summer_or_regular_2, Eng_Nat_spk_2, Course_15
11	8	Summer_or_regular_2, Eng_Nat_spk_2, Course_Instructor_7, Course_11
12	14	Summer_or_regular_2, Eng_Nat_spk_2, Course_2
13	108	Summer_or_regular_2, Eng_Nat_spk_2
14	128	Summer_or_regular_2

A. 3. Kernel Concept Set Analysis of TA Assignments (S_c set to 8%)

Concept ID	Number of TAs Sharing Attributes	Highlighted Attributes
1	**1**	**Class_Size_11, Course_19, Summer_or_regular_2, Eng_Nat_spk_2, Course_Instructor_16**
2	**1**	**Course_Instructor_1, Summer_or_regular_2, Eng_Nat_spk_2, Course_8, Class_Size_18**
3	**1**	**Class_Size_39, Summer_or_regular_2, Course_2, Eng_Nat_spk_2, Course_Instructor_9**
4	**2**	**Course_3, Class_Size_13, Summer_or_regular_1, Eng_Nat_spk_1, Course_Instructor_13**
5	2	Course_3, Summer, Course_Instructor_15, Class_Size_17, Eng_Nat_spk_2
6	2	Class_Size_19, Course_3, Summer, Course_Instructor_23, Eng_Nat_spk_1
7	3	regular, Eng_Nat_spk_2, Course_1, Class_Size_51
8	3	Summer_or_regular_2, Eng_Nat_spk_2, Course_3, Course_Instructor_8
9	3	Summer_or_regular_2, Eng_Nat_spk_2, Course_5, Course_Instructor_9
10	3	Summer_or_regular_2, Course_3, Course_Instructor_22, Eng_Nat_spk_1
11	4	Course_7, Eng_Nat_spk_2, Summer_or_regular_2, Course_Instructor_25
12	4	Summer_or_regular_2, Eng_Nat_spk_2, Course_3, Course_Instructor_23
13	**5**	**Summer_or_regular_2, Eng_Nat_spk_2, Course_3, Course_Instructor_10**
14	6	Eng_Nat_spk_2, Class_Size_20, Summer_or_regular_1, Course_3
15	**7**	**Course_Instructor_18, Eng_Nat_spk_2, Summer_or_regular_2**
16	**7**	**Course_Instructor_13, Eng_Nat_spk_2, Summer_or_regular_2**

DOI: 10.14750/ME.2026.038

17	7	Summer_or_regular_2, Eng_Nat_spk_2, Course_15
18	8	Summer_or_regular_2, Eng_Nat_spk_2, Course_Instructor_7, Course_11
19	14	Summer_or_regular_2, Eng_Nat_spk_2, Course_2
20	**20**	**Summer_or_regular_2, Eng_Nat_spk_1**
21	108	Summer_or_regular_2, Eng_Nat_spk_2
22	128	Summer_or_regular_2

A. 4. List of Generated Concept Lattices

Id	Intent	Frequency
0	1, 2, 7	0.96
1	8, 1, 2, 7	0.89
2	1, 2, 3, 7, 8	0.85
3	1, 3, 7, 8, 9	0.32
4	1, 7	0.68
5	8, 1, 7	0.71
6	8, 1, 3, 7	0.35
7	1, 2, 4, 6	0.78
8	1, 2	0.77
9	1	0.34
10	1, 2, 3, 4, 6	0.63
11	1, 2, 3	0.92
12	1, 3	0.67
13	1, 3, 4, 5	0.75
14	1, 4	0.39
15	1, 3, 4	0.54
16	1, 3, 4, 6	0.47
17	1, 4, 6	0.61